Titre: Title:	A twist decomposition approach to solve functionally-redundant serial manipulators
Auteur: Author:	Liguo Huo
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Huo, L. (2005). A twist decomposition approach to solve functionally-redundant serial manipulators [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7396/

NOTE TO USERS

UNIVERSITÉ DE MONTRÉAL

A TWIST DECOMPOSITION APPROACH TO SOLVE FUNCTIONALLY-REDUNDANT SERIAL MANIPULATORS

LIGUO HUO
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE MÉCANIQUE)
APRIL 2005

Your file Votre référence
ISBN: 0-494-01341-9
Our file Notre référence
ISBN: 0-494-01341-9

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

A TWIST DECOMPOSITION APPROACH TO SOLVE FUNCTIONALLY-REDUNDANT SERIAL MANIPULATORS

présenté par: HUO Liguo

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. GOURDEAU, Richard, Ph.D., président

M. BARON, Luc, Ph.D., membre et directeur de recherche

M. CLOUTIER, Guy, D.Ing., membre

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Luc Baron. He directed me to the domain of robotics and provided the advice, encouragement, and enthusiasm that I needed.

Thanks are also due to my friends Xiaoyu Wang and Sofiane Achiche, who shared with me their vast knowledge and experience on robotic manipulators and assisted me in writing papers.

I would like to dedicate this thesis to my parents, who have always encouraged me to do my best throughout my life.

I also wish to thank my wife. I would not have been able to finish this thesis without her support.

RÉSUMÉ

Cette thèse propose une nouvelle approche de résolution de la redondance pour des tâches robotiques exigeant moins de six degré-de-liberté (DDL). Cette approche est basée sur la décomposition du torseur de vitesse dans deux sous-espaces cartésien. Au lieu de projeter la solution sur le noyau de la matrice jacobienne comme le fait la majorité des autres approches, nous décomposons le torseur de vitesse associé à l'effecteur en deux sous-espaces appropriés, l'un est le sous-espace de la tâche principale, l'autre celui de la tâche redondance. La résolution de la redondance est donc optimisée dans le sous-espace redondant du torseur. Cette approche peut être appliquée à toutes les tâches exigeant moins de six DDL quelque soit le nombre de DDL du manipulateur.

De plus, deux méthodes de calcul des projections sur le noyau de la matrice jacobienne sont également étudiées ici. Le premier algorithme emploie l'inverse généralisée de la matrice jacobienne, le deuxième applique des réflexions de Householder sur la matrice Jacobienne. En comparant numériquement les deux algorithmes, l'algorithme des réflexions de Householder peut atteindre une solution beaucoup plus précise parce qu'il évite d'amplifier le conditionnement de la matrice jacobienne.

Afin d'éviter les limites articulaires en soudage à l'arc, notre approche a été mise en application et comparée numériquement à l'approche augmentée, les deux utilisent l'algorithme des réflections de Householder. L'approche augmentée nécessite d'ajouter une articulation virtuelle au manipulateur afin de rendre la matrice jacobienne sous-déterminée. Les résultats de simulation ont montré que notre approche permet d'atteindre des trajectoires plus douces et plus précises. Si elle est mise en application sur un manipulateur à 6 DDL, la pseudo-inverse de la jacobienne est évitée, et le processus de calcul devient alors simple.

Un environnement de programmation et d'analyse hors-ligne sont présentés dans la

deuxième partie de ce mémoire. Afin de résoudre la redondance pour un manipulateur réel, le système peut produire automatiquement des programmes robot pour la tâche cartésienne requise. Dans ce système, un programme de prétraitement résout le problème géométrique inverse lié à la tâche désirée dans l'espace cartésien, puis un simulateur graphique montre le mouvement du manipulateur afin de vérifier si la trajectoire répond ou non aux exigences. Finalement, un module de traduction de langage convertit la trajectoire dans le langue du robot (programme en VAL II). Ce programme est transmis directement au robot (PUMA560). En employant ce système, le programme de contrôle du robot peut être examiné et mis au point avant qu'il soit exécuté sur le robot réel, et l'exécution des trajectoires complexes devient ainsi possible.

ABSTRACT

This thesis proposes a new redundant-resolution approach to solve robotic tasks requiring less than six degrees-of-freedom (DOF). This approach is called Twist Decomposition Approach. Instead of projecting an optimization criterion onto the null space of the Jacobian matrix as most of the redundancy-resolution schemes do, *twist decomposition approach* firstly decomposes the Cartesian twist of the end-effector into two suitable subspaces; one being the subspace where the main task undergoes, while the other one being the redundant subspace. The redundancy resolution is optimized on the redundant subspace of the twist. Our approach could be applied to all redundant tasks requiring less than six DOF, regardless of how many DOF the manipulator has.

Moreover, two numerical implementations of the projection onto the null space of the Jacobian matrix have been studied. The first algorithm uses the generalized inverse of the Jacobian matrix; while the other one uses Householder reflections on the Jacobian matrix. From the numerical comparison of the two algorithms, Householder reflection algorithm can reach a much higher accuracy level because it avoids amplifying the condition number of the Jacobian matrix, while solving the linear algebraic system at hand.

In order to avoid joint-limits in arc-welding, *twist decomposition approach* was implemented and compared numerically with the augmented approach. The augmented approach adds virtual joints to the manipulator in order to reach an undetermined linear algebraic system with at least one-DOF of redundancy. The results show that *twist decomposition approach* reaches a smoother and more accurate trajectory. If implemented on a six DOF manipulator, the pseudo-inverse of the Jacobian matrix is avoided, and the computation process is simplified.

Besides the proposition of the new redundancy-resolution approach, this thesis also attempts to apply the theory to reality. An off-line programming and analysis envi-

ronment is presented in second part of the thesis. In order to make the redundant-resolution scheme implementable on a real manipulator, the system could automatically generate the robot-understandable trajectory from the Cartesian task space. In this environment, a preprocessing program solves the inverse kinematics problems related to the desired task in a Cartesian space, then a graphic simulator displays the motion of the manipulator in order to verify whether the trajectory meets the requirements or not. Finally a translator module converts the joint space trajectory into a robot language (VAL II) program. This program is transferred into robot (PUMA560) directly. By using this system, the robot controlling program can be tested and debugged before it is executed on the real robot, and the implementation of a complicated joint space trajectory on a real robot becomes possible.

CONDENSÉ EN FRANÇAIS

0.1 Introduction

Dans ce mémoire, les sources de redondance cinématique d'un manipulateur sériel sont classées dans deux catégories: la redondance intrinsèque et la redondance fonctionnelle. Ces redondances sont définies au chapitre 1 (voir la définition 1.1, 1.2 et 1.3 en pages 6 et 9).

Reliées à la redondance intrinsèque, les méthodes de résolution de redondance (RR) sont classifiées en méthodes optimales globale et locale. Le problème de RR a été également résolu au niveau des déplacements et des vitesses. La plupart des chercheurs ont travaillé au niveau des vitesses et ont employé l'inverse généralisée de Moore-Penrose ou l'inverse généralisée pondérée de la matrice Jacobienne. Les méthodes employant l'inverse généralisée utilisent soit la solution de norme minimale (voir eq.(2.35)) soit la solution de norme non-minimale (voir eq.(2.37)). La solution de norme non-minimale ajoute une composante homogène à la solution de norme minimale. La composante homogène projette un vecteur arbitraire sur le noyau (ou espace nul) de la matrice Jacobienne. L'équation (2.37) est largement utilisée par les chercheurs pour résoudre des tâches redondantes. Arenson, Angeles et Slutski (1998) ont proposé d'employer la réflexion de Householder dans la solution de norme non-minimale. L'utilisation de la réflexion de Householder évite d'amplifier le conditionnement de la matrice Jacobienne et donc l'erreur de calcul de la solution.

Dans le cas de la redondance fonctionnelle, les méthodes de RR intrinsèques travaillant dans l'espace nul de la matrice Jacobienne ne peuvent être directement employées parce que la dimension de l'espace nul de la Jacobienne est égale à zéro.

Afin d'obtenir un système sous-déterminé, il y a deux possibilités: augmenter la dimension du vecteur vitesse articulaire, ou réduire la dimension du torseur de vitesse de l'effecteur. Correspondant aux deux possibilités, les méthodes de RR fonctionnelle peuvent être classifiées en deux groupes: l'approche augmentée et l'approche réduite. La méthode de l'articulation virtuelle selon l'approche augmentée et la méthode d'élimination selon l'approche réduite sont passées en revue au chapitre 2.

Dans ce mémoire, on propose une nouvelle méthode de RR fonctionnelle qui est basée sur la décomposition du torseur de vitesse. Cette méthode ne nécessite pas la projection sur l'espace nul de la matrice Jacobienne. La méthode de décomposition du torseur de vitesse a une applicabilité potentielle à toutes les tâches de moins de six DDL quelque soit le nombre de DDL du manipulateur.

Afin de mettre en application la méthode de RR à la commande des manipulateurs redondants, un système de programmation hors-ligne, appelé PUMASIM a été développé. PUMASIM inclut quatre modules: le module de prétraitement; le module graphique de simulation; le module de traduction de langage robot; et le module de communication. Ce système permet à l'opérateur de créer une trajectoire dans l'espace opérationnel à l'aide d'un simulateur graphique sans nécessiter l'écriture des codes du langage de programmation du robot.

0.2 La méthode de décomposition du torseur de vitesse

Tous les vecteurs de $\mathbb{R}^3$ peuvent être décomposés en deux parties orthogonales en utilisant le projeteur $\mathbf{M}$ et son complément orthogonal $\mathbf{M}^{\perp}$. Les projeteurs sont donnés pour les quatre dimensions possibles du sous-espace de $\mathbb{R}^3$: de la tâche de dimension zéro à la tâche de 3 dimensions (voir eq.(3.2)).

Selon la décomposition orthogonale des vecteurs, n'importe quelle rangée du torseur de l'espace $2 \times \mathbb{R}^3$ peut également être décomposée en deux parties orthogonales en utilisant le projeteur $\mathbf{T}$ et son complément orthogonal $\mathbf{T}^{\perp}$. Les projeteurs de torseurs de vitesse sont définis par une matrice diagonale de deux projeteurs des vecteurs de $\mathbb{R}^3$ (voir l'eq.(3.6)). Pour les manipulateurs séries redondants fonctionnellement, il est possible de décomposer le torseur de vitesse de l'effecteur en deux parties orthogonales dans le sous-espace tâche et le sous-espace redondant. Par conséquent, la décomposition orthogonale du torseur de vitesse peut se substituer à la solution de norme minimale de la cinématique inverse du manipulateur redondant (voir l'eq.(3.9)). La première partie du côté droit de l'eq.(3.9) atteint le déplacement articulaire exigé par la tâche, alors que la deuxième partie atteint le déplacement articulaire dans le sous-espace redondant. L'équation (3.9) est la contribution originale principale de ce mémoire, elle n'exige pas la projection sur l'espace nul de la matrice Jacobienne comme la majorité des algorithmes de résolution de redondance, mais exige plutôt une projection orthogonale basée sur la géométrie instantanée de la tâche à accomplir.

Puisque les deux projeteurs de vecteur dans la matrice de projeteur de torseur ont quatre dimensions possibles dans $\mathbb{R}^3$, le projeteur de torseur a 16 possibilités comme montré au Tableau 3.1. Selon la condition de la tâche, un projeteur différent du torseur sera choisi pour décomposer le torseur. L'algorithme 3.1 montre la résolution de redondance fonctionnelle avec la méthode de décomposition du torseur de vitesse.

0.3 Application au soudage à l'arc

En effectuant des opérations de soudage à l'arc, l'électrode possède un axe de symétrie qui peut être tourné sans interférer avec la tâche à exécuter. Si nous ne ti-

rons pas profit de l'axe de symétrie de l'électrode, la tâche de soudage à l'arc mise en application par un robot à six articulations n'est pas une tâche redondante. La méthode de l'articulation virtuelle et la méthode de décomposition du torseur de vitesse tirent profit de l'axe de symétrie de l'électrode comme mouvement non pertinent, et la tâche devient une tâche fonctionnellement redondante. Un robot PUMA 560 muni d'une électrode de soudage doit accomplir une tâche de soudage d'un tuyau sur une bride exigeant l'exécution de plusieurs trajectoires circulaires consécutives. Afin de créer une trajectoire continue dans l'espace articulaire, la trajectoire dans l'espace de tâche est interpolée à partir d'un ensemble de poses discrètes, et le problème de cinématique inverse est résolu à chaque pose échantillon. Trois méthodes différentes de solution du modèle géométrique inverse sont employées, soit: méthode sans RR; méthode de l'articulation virtuelle et méthode de décomposition du torseur de vitesse. Comme illustré à la Fig. 4.5, la trajectoire articulaire sans l'utilisation de la méthode de RR n'est pas réalisable sans dépasser les limites articulaires dès le deuxième tour. Dans les mêmes conditions, la méthode de l'articulation virtuelle et la méthode de décomposition du torseur de vitesse sont capables de produire une trajectoire articulaire sans dépasser les limites articulaires pour la tâche de plusieurs tours consécutifs (voir les Figs. 4.6 et 4.7).

Cependant, des vitesses articulaires excessives apparaissent à chaque tour avec la méthode de l'articulation virtuelle. Ces vitesses rendent l'exécution de la trajectoire articulaire impossible. Les trajectoires articulaires obtenues par la méthode de décomposition du torseur de vitesse sont plus lisses et précises que celles obtenues par la méthode de l'articulation virtuelle.

0.4 PUMASIM: un système de programmation hors-ligne

Comme montré à la Fig. 5.1, PUMASIM comporte les modules suivants:

- le module de prétraitement qui produit la trajectoire dans l'espace articulaire

et vérifie les singularités;

- le module graphique de simulation;
- le module de traduction en langage robot;
- le module de communication entre le contrôleur du robot et un micro-ordinateur.

Dans ce système, après que la trajectoire désirée de l'effecteur dans l'espace opérationnel soit décrite, un fichier de données de la position articulaire le long de la trajectoire est créé. Ce module de prétraitement, développé en MATLAB, résout le modèle géométrique inverse à chaque position de l'effecteur et calcule la configuration correspondante du robot dans l'espace articulaire. Des poses singulières peuvent être détectées et les vitesses articulaires peuvent être calculées par le module de prétraitement. Une fois la trajectoire dans l'espace articulaire produite, le mouvement du manipulateur peut être montré graphiquement avec le module de simulation, et la trajectoire du manipulateur est ainsi vérifiée pour répondre à nos exigences. Si la trajectoire répond à nos exigences, une traduction prenant en considération le langage de robot produit les codes correspondants du programme en VAL II. Un programme en VAL II peut être chargé dans le contrôleur par liaison sérielle entre le contrôleur du robot et le terminal avec le module de communication. En résumé, la tâche du robot est définie dans l'espace opérationnel, mais la trajectoire dans l'espace articulaire est mise en application dans le contrôleur du robot. Ce système de programmation hors-ligne intégré a été mis en œuvre pour le PUMA 560, mais il pourrait être employé avec d'autres architectures de robots sériels avec quelques modifications.

En employant ce système de programmation hors-ligne, un chemin circulaire dans l'espace cartésien est mis en application par le PUMA 560 avec un axe de symétrique de l'effecteur (voir Fig. 5.6). La méthode sans RR et la méthode de décomposition du torseur de vitesse sont appliquées, respectivement. Lorsque la tâche n'est pas

mise en application par la méthode de RR, le module de prétraitement détecte des positions articulaires hors limites et arrête le processus. Cependant, le module de prétraitement avec la méthode de décomposition du torseur de vitesse peut atteindre une trajectoire articulaire optimisée à l'intérieur des limites articulaires et maintenir un bon conditionnement. Un programme en Val II correspondant à la trajectoire articulaire a été chargé dans le contrôleur du PUMA 560 et est mis en œuvre avec succès comme montré à la Fig. 5.11.

0.5 Conclusion

La méthode de décomposition du torseur de vitesse peut atteindre une trajectoire plus lisse et plus précise que la méthode de l'articulation virtuelle pour notre tâche. Bien que la méthode de décomposition du torseur de vitesse puisse seulement résoudre des problèmes de redondance fonctionnelle, elle reste néanmoins significative et intéressante puisqu'un nombre élevé de tâches fonctionnellement redondantes existe dans le milieu industriel, telle que la soudure, la pulvérisation, le fraisage, le découpage par jet d'eau et le découpage au laser, etc.

Dans ce mémoire, la méthode de décomposition du torseur de vitesse est seulement utilisée pour éviter le problème posé par les limites articulaires. Elle exige plus d'étude sur d'autres applications, comme par exemple éviter les obstacles et les singularités.

Pendant le processus d'optimisation, nous notons que les choix de posture initiale et de vecteur pondération affecte considérablement la trajectoire optimisée. Un mauvais choix peut même causer l'échec total de l'optimisation. En fait, ces choix se fondent, la plupart du temps, sur l'expérience. Il est donc nécessaire de développer une méthode plus efficace et plus systématique. Dans le futur, ce sujet de recherche pourra intégrer la solution de RR avec les méthodes de contrôle intelligents, telles

que la logique floue, les réseaux des neurones et les algorithmes génétiques.

CONTENTS

LIST OF FIGURES

LIST OF NOTATIONS AND SYMBOLS

DDL: degré de liberté;

DH: Denavit-Hartenberg;

DKP: Direct Kinematics Problem;

DOF: degrees of freedom;

EE: end-effector;

GI: Generalized Inverse;

GPM: Gradient Projection Method;

GS: Graphics Simulation;

IKP: Inverse Kinematics Problem;

RR: Redundancy-Resolution;

$\mathbf{J}$: Jacobian matrix;

$\mathbf{A}$: the upper three rows of $\mathbf{J}$;

$\mathbf{B}$: the lower three rows of $\mathbf{J}$;

$\boldsymbol{\theta}$: joint position vector;

$\dot{\boldsymbol{\theta}}$: joint velocity vector;

$\mathbf{t}$: twist array expressing the end-effector velocity;

${}^{\mathcal{F}_j}\mathbf{A}_{\mathcal{F}_i}$: homogeneous transformation matrix describing frame $\mathcal{F}_i$ with respect to frame $\mathcal{F}_j$;

$\mathbf{p}$: position vector in Cartesian space;

$\mathbf{Q}$: rotation matrix between two frames;

vect: function transforming a 3×3 rotation matrix into axial vector;

DKP: function solving DKP;

IKP: function solving IKP;

JACOB: function computing jacobian matrix;

$\mathcal{J}$: joint space;

$\mathcal{T}$: task space;

$\mathcal{O}$: operational space;

n: dimension of the joint space;

o: dimension of the operational space;

t: dimension of the task space;

r_I: degree of intrinsic redundancy;

r_F: degree of functional redundancy;

r_K: degree of kinematic redundancy;

$\mathbf{W}$: positive-definite weighting matrix;

$\mathbf{M}$: projector;

$\mathbf{P}$: plane projector;

$\mathbf{L}$: line projector;

$\mathbf{T}$: twist projector;

$\overline{e}_e$: the mean value of orientation errors;

$\overline{e}_p$: the mean value of position errors;

$\overline{o}_e$: the mean value of orientation errors after projected;

LIST OF TABLES

CHAPITRE 1

INTRODUCTION

1.1 Background and Basic Terminology

A *manipulator* is a device that helps human beings to perform manipulating tasks. A *robotic manipulator* is to be distinguished from the previous for its ability to lead itself through computer control. Once programmed, it can implement the same task repeatedly. In general, robotic manipulators can be studied using the concept of *kinematic chain.* A kinematic chain is a set of rigid bodies, also called links, coupled by *kinematic pairs.* A kinematic pairs is the coupling of two rigid bodies so as to constrain their relative motion. There are two basic types of kinematic pairs, namely, *upper* and *lower* kinematic pairs. An upper kinematic pair is obtained through either line contact or point contact, and thus, appear in cam-and-follower, gear trains, and roller bearings, for example. A lower kinematic pair occurs when contact takes place along a surface common to the two bodies[1]. As shown in Fig. 1.1, there are six lower kinematic pairs, namely, (E) Planar, (S) Spherical, (C) Cylindrical, (R) Revolute, (P) Prismatic, and (H) Helicoidal.

As shown in Fig. 1.2 and classified in Table 1.1, kinematic chains are termed either *simple* or *complex*; and either *open* or *closed.* Simple chains are those having links with a *degree of connectivity*[1] of less than or equal to two, while complex chains are those having at least one link with a degree of connectivity greater than two. Open and closed simple kinematic chains occur in *serial manipulators* and *linkages*,

[1]The *degree of connectivity* of a body is defined as the number of bodies directly connected to the said body through kinematic pairs.

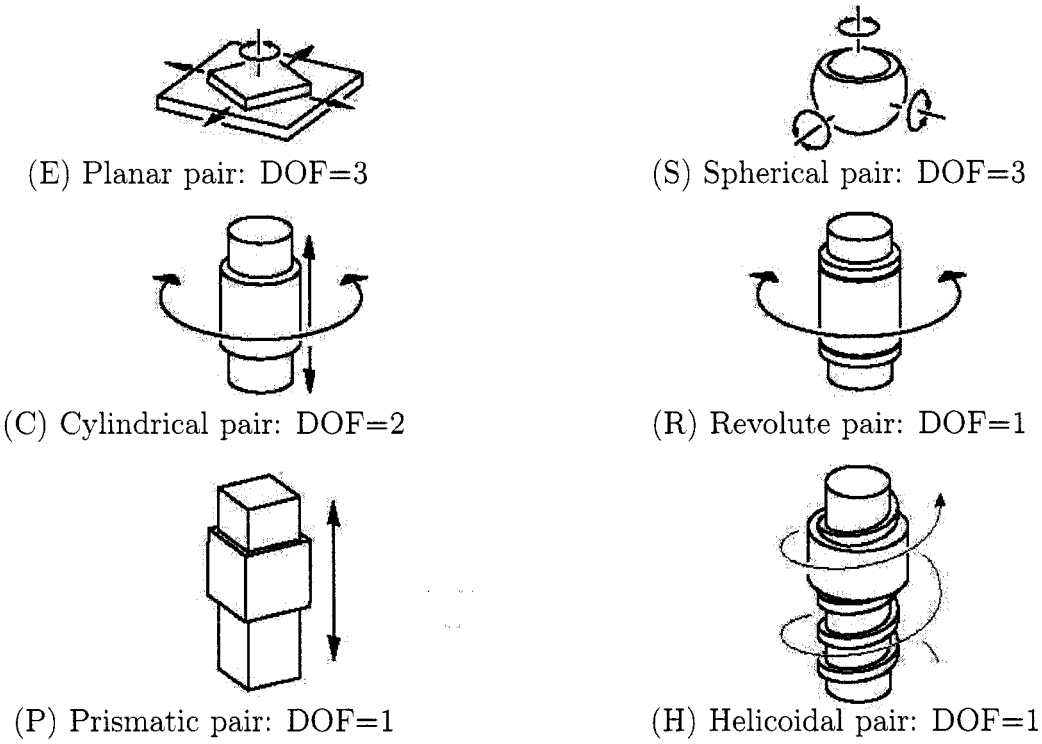


Figure 1.1 The six lower kinematic pairs

respectively. Open complex kinematic chains are sets of open kinematic chains in a tree-type structure, and occur therefore in tree-type manipulators. Closed complex kinematic chains are termed either *parallel* or *hybrid* depending on whether the kinematic loops lie in parallel arrays or not. Closed complex kinematic chains with loops in parallel arrays occur in *parallel manipulator*, while all the other kinematic chains occur in *hybrid manipulators*, *i.e.*, those containing either more than two links with a degree of connectively greater than two, or those containing both closed and open kinematic chains. Figure 1.3 presents four examples of different types of manipulators, which include a serial manipulator, a parallel manipulator, a robotic hand and a walking machine. Clearly, both the robotic hand and the walking machine can be classified as tree-type manipulators.

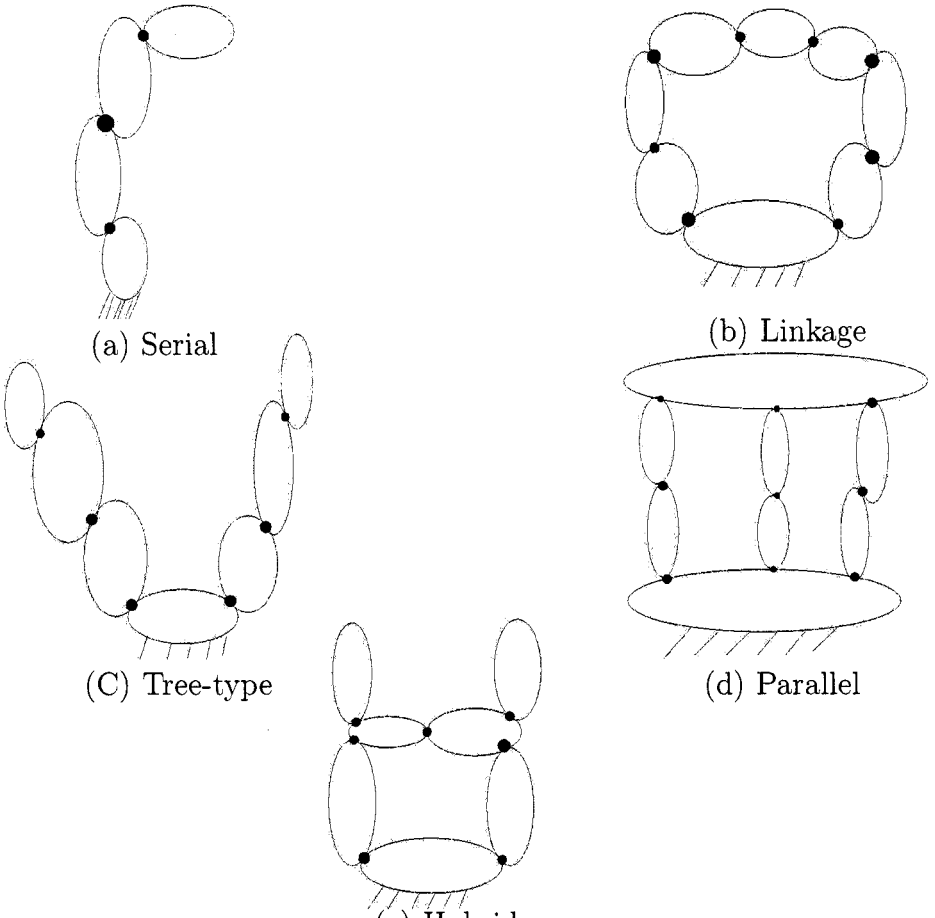


Figure 1.2 Classification of the kinematic chains

In this thesis, we focus on serial manipulators, *i.e.*, simple open kinematic chains. In such manipulators, there are exactly two bodies with a degree of connectivity of one, called end-bodies, and all the other bodies with a degree of connectivity of two. One end-body is arbitrary regarded as fixed and is called the *base*, while the other end-body is regarded as movable and is called the moving body, or the *end-effector* (EE) of the manipulator.

	Open	Closed
Simple $C_{max} \leq 2$	Serial Manipulator $C_i = 1,\ i = 1, 2$ $C_i = 2,\ i > 2$	Linkage $C_i = 2$
Complex $C_{max} > 2$	Tree-type manipulator $C_{max} > 2$ with no loop	Parallel manipulator $C_i > 2,\ i = 1, 2$ $C_i = 2,\ i > 2$
		Hybrid manipulator all other cases

Table 1.1 Classification of the kinematic chains. (C_i is the degree of connectivity of link i, while C_{max} is the maximum degree of connectivity of all links)

1.1.1 Degrees of Freedom of a Mechanical System

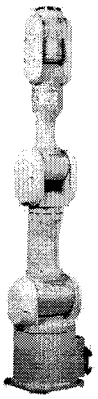
In general, the minimum number of variables (also called coordinates) to completely specify the configuration of a mechanical system is called the degrees of freedom (DOF) for that system. For a serial manipulator, each independent variables is typically associated with a joint i, *e.g.*, θ_i. Since, the DOF related to a serial manipulator is the sum of DOF of each joint. The combination of the joint positions of a manipulator is referred to as a *posture*. We can combine the joint position into a joint vector, namely $\boldsymbol{\theta}$, given by

$$\boldsymbol{\theta} \equiv [\theta_1 \cdots \theta_n]^T \in \mathcal{J}^n, \tag{1.1}$$

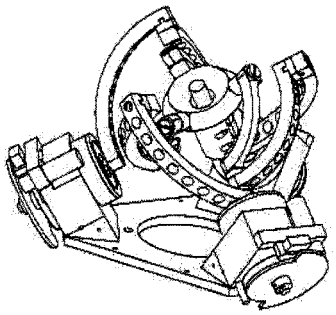
where $\mathcal{J}$ is the *Joint space* in which $\boldsymbol{\theta}$ is defined, and its dimension n is given as, $n = dim(\mathcal{J})$, is therefore the DOF of the manipulator.

1.1.2 Degrees of Freedom of the End-Effector

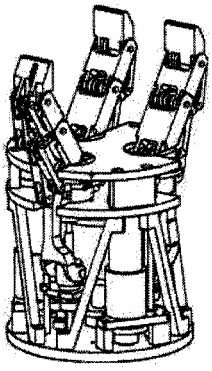
The combination of the position and the orientation of a rigid body is referred to as the *pose* of the corresponding body. For a rigid body freely moving in three-



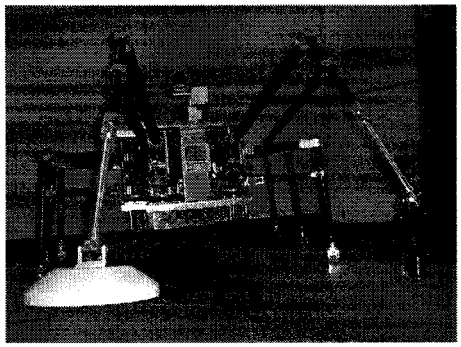
a. Serial manipulator



b. Parallel manipulator



c. Robotic hand



d. Walking machine

Figure 1.3 A Classification of the robotic manipulators: (a) PA10-6C from MITSUBISHI[2]; (b) Agile Eye from Laval University[3]; (c)SARAH Robotic Hand from Laval University[4]; (d)COMET-II from Chiba University[5].

dimensional space, a minimum of six coordinates are required to completely define its mobility or degrees of freedom (DOF). At the displacement level, the pose of a rigid body can be defined by the position of a point of the body together with the orientation of the body around that point. The position of a point of the body can be defined by specifying its three Cartesian coordinates in some convenient coordinate frame, *e.g.*, p_x, p_y and p_z. Similarly, the orientation of the body around that point can be defined by three angles in some convenient coordinate frame, *e.g.*, θ_x, θ_y and θ_z. We can combine the six elements into one array, namely $\mathbf{x}$,

given as

$$\mathbf{x} \equiv \begin{bmatrix} \theta_x & \theta_y & \theta_z & p_x & p_y & p_z \end{bmatrix}^T. \tag{1.2}$$

The motion of the EE can be defined by the motion of $\mathbf{x}$. The space in which the EE undergoes its motion is usually called the *operational space*, denoted by $\mathcal{O}$, and its dimension o is given as $o = dim(\mathcal{O})$.

For a specific task, the motion of the EE may require the whole operational space $\mathcal{O}$ or only a subspace of $\mathcal{O}$. In both cases, we call the space in which the task is undergoing, the *task space* $\mathcal{T}$, its dimension t is given as $t \leq o$, since $dim(\mathcal{T}) \leq dim(\mathcal{O})$. In all cases, we must have $n \geq o \geq t$ with $\mathcal{T} \subseteq \mathcal{O}$, otherwise the task can not be performed by the manipulator.

1.1.3 Kinematic Redundancy

The research domain of this thesis is limited to kinematic redundancy.

Definition 1.1: Kinematic redundancy

A pair made of a serial manipulator and a task is said to be kinematically redundant when the dimension of the joint space $\mathcal{J}$, denoted by $n = \dim(\mathcal{J})$, is greater than the dimension of the task space $\mathcal{T}$ of the EE, denoted by $t = \dim(\mathcal{T}) \leq 6$, while the task space being totally included into the resulting operation space of the manipulator, i.e., *$\mathcal{T} \subseteq \mathcal{O}$, and hence, $n > t$. The degree of kinematic redundancy of a pair of serial manipulator-task, namely r_K, is computed as*

$$r_K = n - t. \tag{1.3}$$

In a system with kinematic redundancy, it is possible to change the internal struc-

ture or configuration of the mechanisms without changing the position and orientation of the EE[6]. In this thesis, the term "redundancy" refers to "kinematic redundancy".

A typical example of a redundant manipulator is the human arm, which has seven DOF from the shoulder to the wrist. If the base and the hand position and orientation are both fixed, requiring six DOF; the elbow can still be moved, due to the additional mobility associate with the redundant DOF. Thus, it becomes possible to avoid obstacles in the workspace. Furthermore, if a joint of a redundant manipulator reaches its mechanical limit, there might be other joints that allow execution of the same prescribed EE motion.

1.2 Problem Formulation

Many redundant manipulators have been developped so far. Some were developed for research purposes, while others have already been used in real applications. The Canadarm2, that plays a key role in space station assembly and maintenance, is a seven-DOF manipulator as shown in Fig. 1.4. This manipulator has three joints as a shoulder, one joint as elbow and three joints as a wrist. Because there is at least one degree of redundancy, the Canadarm2 can change configuration without moving its hand.

However, the control of a redundant manipulator is more challenging, since there are infinitely many joint trajectories that exist for a given task. The operator must evaluate the best one according to a performance criterion. The solution strategy, which effectively exploits the potential advantages of kinematically redundant mechanisms, is called: *Redundancy Resolution* (RR) scheme. Since there is no general RR scheme solution, RR scheme has attracted the attention of many

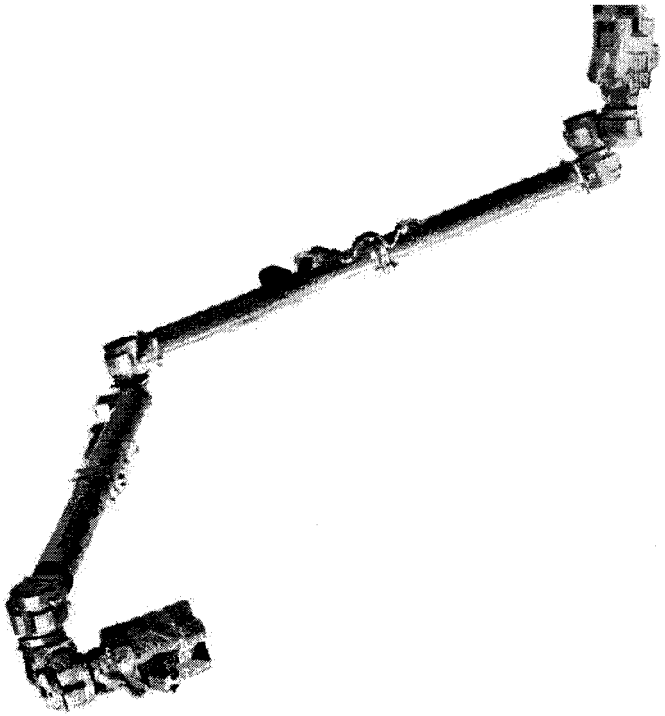


Figure 1.4 The Canadarm2: a 7-joint redundant robot [7]

researchers for three decades.

1.3 Research Motivation

Based on the definition of kinematic redundancy, it is a concept related to a given task. Even if a manipulator is kinematic redundant for a specific task, it may not be redundant for another task. Hence, according to the relation among joint space, operational space and task space, kinematic redundancy can be classified into two groups, *e.g.*, *functional redundancy*, and *intrinsical redundancy.*

Definition 1.2: Intrinsic redundancy

A serial manipulator is said to be intrinsically redundant when the dimension of the joint space $\mathcal{J}$, denoted by $n = \dim(\mathcal{J})$, is greater than the dimension of the

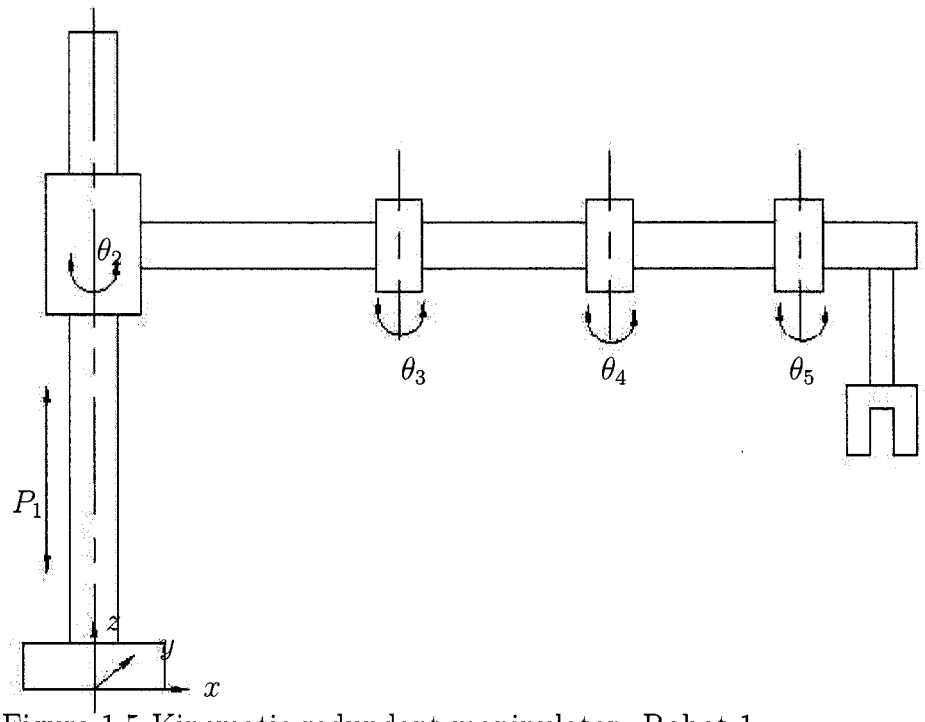


Figure 1.5 Kinematic redundant manipulator: Robot 1

resulting operational space $\mathcal{O}$ of the EE, denoted by $o = \dim(\mathcal{O}) \leq 6$, i.e., *$n > o$. The degree of intrinsic redundancy of a serial manipulator, namely r_I, is computed as*

$$r_I = n - o \tag{1.4}$$

Definition 1.3: Functional redundancy

A pair made of a serial manipulator and a task is said to be functionally redundant when the dimension of the operational space $\mathcal{O}$ of the EE, denoted by $o = \dim(\mathcal{O}) \leq 6$, is greater than the dimension of the task space $\mathcal{T}$ of the EE, denoted by $t = \dim(\mathcal{T}) \leq 6$, while the task space being totally included into the operation space of the manipulator, i.e., *$\mathcal{T} \subseteq \mathcal{O}$, and hence, $o > t$. The degree of functional*

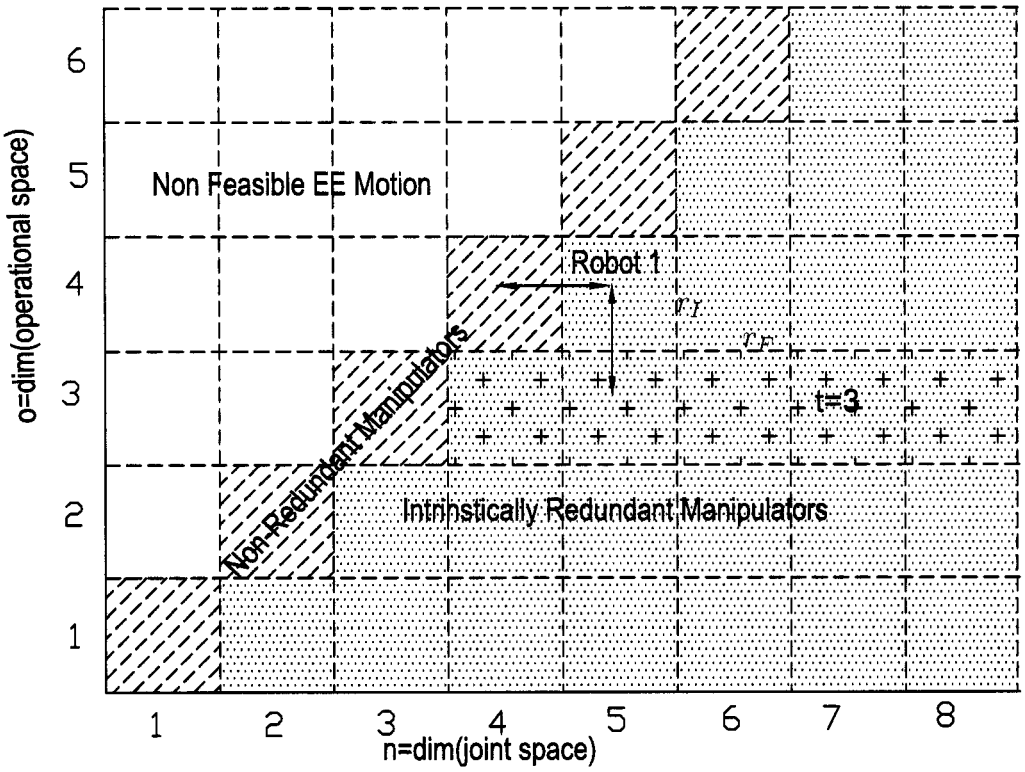


Figure 1.6 Intrinsic and functional redundancies of serial robotic tasks

redundancy of a serial manipulator-task pair, namely r_F*, is computed as*

$$r_F = o - t. \tag{1.5}$$

Apparently from eqs.(1.4) and (1.5), the kinematic redundancy of eq.(1.3) can be rewritten as

$$r_K = r_I + r_F, \tag{1.6}$$

which makes clear that kinematic redundancy comes from two different sources: the functional redundancy and the intrinsic redundancy. The distinction among the different redundant manipulators is shown in Fig. 1.6. As $n = o$, these are non-redundant manipulators; as $n > o$, intrinsic redundancy occurs. In the case of $n < o$, the manipulator does not exist because it can not fulfill the motion in

operational space. For example, let us consider a 4-R-1-P serial manipulator as shown in Fig. 1.5. For this manipulator, $\mathcal{J}$ is of dimension 5, *i.e.*, $\dim(\mathcal{J}) = n = 5$, the resulting $\mathcal{O}$ is only of dimension 4 (positioning a point of the EE in 3D space and orienting the EE around an axis only), *i.e.*, $\dim(\mathcal{O}) = o = 4$, and hence, this manipulator has a degree of intrinsic redundancy of one, *i.e.*, $r_I = n - o = 5 - 4 = 1$. For the positioning task in 3D space without considering its orientation , $\mathcal{T}$ is only of dimension 3, *i.e.*, $\dim(\mathcal{T}) = t = 3$, and hence, the degree of functional redundancy of the pair of manipulator-task is one, *i.e.*, $r_F = o - t = 4 - 3 = 1$. Finally, the kinematic redundancy of this pair of manipulator-task is two because $r_K = r_I + r_F = 1 + 1 = 2$, and not one. In the literature, most of the research works on redundancy-resolution of serial manipulators studied the case of $r_F = 0$, and thus, $r_K = r_I$. In this thesis, we will study the opposite case, *i.e.*, $r_I = 0$ and thus, $r_K = r_F$.

In 3D space, seven-axes serial manipulators are intrinsically redundant, *e.g.* the Canadarm2. For intrinsically redundant manipulator, the scheme of using the null space[2] of the Jacobian matrix can be directly used to select an optimized solution.

Six-axes serial manipulators are the most well known and popular robots because they are multipurpose. Moreover, they are able to be functionally redundant in 3D space. As shown in Figs. 4.10 and 1.8, arc-welding, laser cutting and milling tasks can be performed with six-axes manipulators. Because the rotation along the EE symmetry axis are irrelevant to the completion of the tasks ($t = 5$), the manipulators are functionally-redundant. A similar situation occurs for the pick and place operation of the axis-symmetric objects ($t = 4$ or 5).

Most of the redundancy resolutions focus on the solution of intrinsically-redundant manipulators, and use the null-space of the Jacobian matrix. However, the RR

[2]Null space: If $\mathbf{T}$ is a linear transformation of $\mathbb{R}^n$, then the null space $\mathbf{Null(T)}$ is the set of all vectors $\mathbf{x}$ such that $\mathbf{T(x)} = \mathbf{0}$, *i.e.*, $\mathbf{Null(T)} \equiv \{\mathbf{x} : \mathbf{T(x)} = \mathbf{0}\}$.[8]

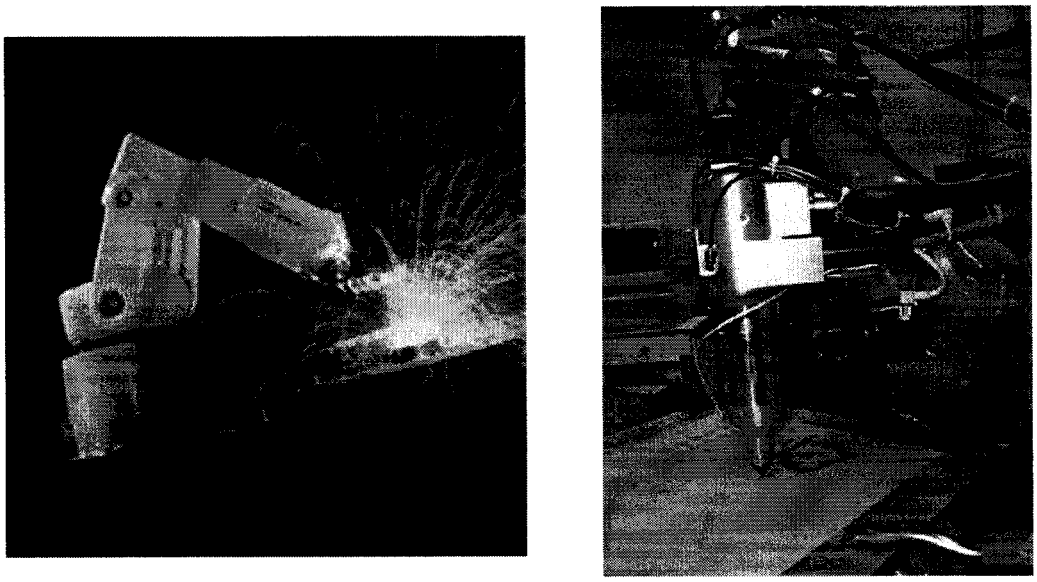


Figure 1.7 Arc welding task (left) and laser cutting task (right)

schemes that use the null space of the Jacobian matrix can not directly be used to solve functionally-redundant problem. In this cases, the Jacobian matrix is a full rank square matrix, and hence the dimension of the null space of the said Jacobian matrix is zero.

This thesis focuses on studying several general solution schemes of functionally redundant manipulators, and proposes a new RR scheme, namely the *twist decomposition method*, that does not working on the null space of the Jacobian matrix. The twist decomposition method has a potential applicability to all less than six-DOF redundant tasks, independently if the task is implemented by a functionally or intrinsically redundant manipulator.

In order to implement our RR scheme to the real control of redundant manipulators, an off-line programming system has been developed. The whole controlling process, in which the RR scheme is applied, is shown in Fig. 1.9.

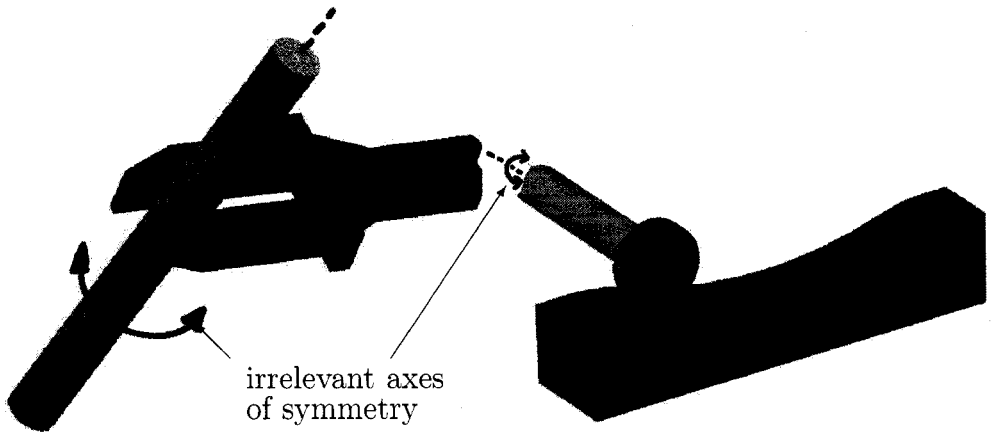


Figure 1.8 Pick-up task (left) and milling task (right)

1.4 Thesis Organization

This thesis has six chapters. Chapter 2 introduces previous works in the field of the redundancy resolution and continuous trajectory planning. Chapter 3 shows the development of twist decomposition method, which is able to handle functionally-redundant tasks. In chapter 4, the twist decomposition method is applied on arc-welding task with joint-limits avoidance problem, and is compared with *virtual joint method* that works on the null space of the Jacobian matrix. In chapter 5, an off-line programming and analysis system for a general robotic manipulator is described. The system makes the implementation of the redundant task in real robots possible. Finally, chapter 6 concludes the thesis.

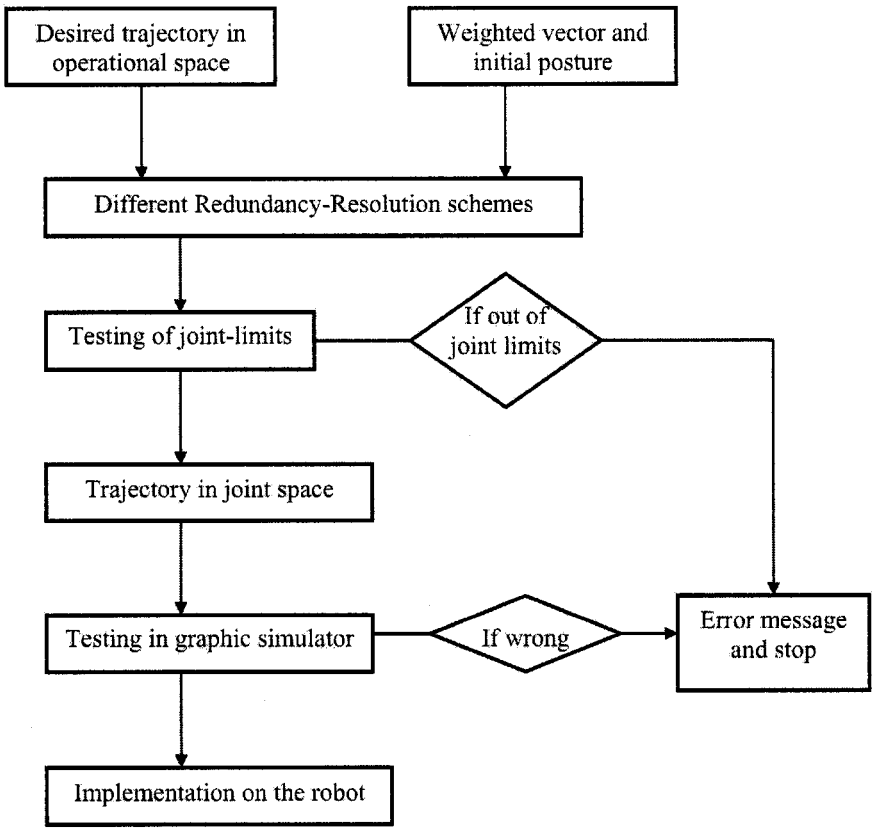


Figure 1.9 Flowchart of the redundant manipulator controlling process

CHAPITRE 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents some previous works on redundant robotic manipulators and trajectory planning. The basic concepts of the kinematics of robotic manipulators are recalled, in particular the serial ones, since these are fundamentals of the following discussion. Then, the trajectory planning and programming are introduced, including continuous path planning and robot programming. Finally, this chapter focus on kinematic redundancy and review some early works of this field of research.

2.2 Kinematics of Serial Manipulators

2.2.1 Denavit-Hartenberg Parameters

Denavit and Hartenberg[9] (DH) established a well-known method for developing the kinematic model of serial manipulators, which is based on homogeneous transformation matrices. Let us attach frame $\mathcal{F}_i$ to link i, with the z_i axis along the axis of joint $i+1$, and the x_i axis aligned along the common perpendicular to the axes of joints i and $i+1$. The geometry of each link is described by a set of four parameters, θ_i, r_i, l_i and α_i. The joint variable is either, θ_i for a revolute joint, or r_i for a prismatic joint. The other parameters are known constants for a given link geometry. For example, l_i is the distance along the common perpendicular of

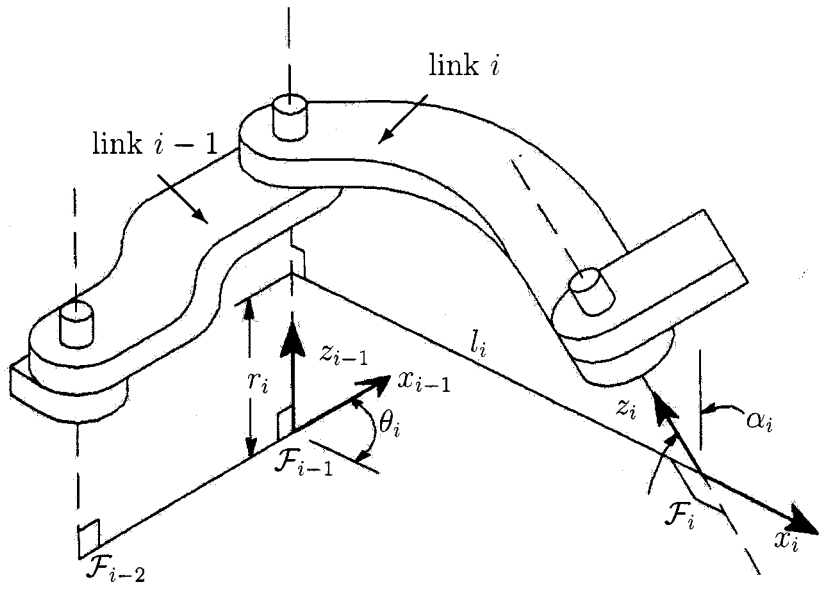


Figure 2.1 Denavit-Hartenberg parameters (figure from [10])

two consecutive joint axes, while α_i is the angle of rotation between these two axes around the common perpendicular. Finally, the position and orientation of frame $\mathcal{F}_i$ with respect to frame $\mathcal{F}_{i-1}$ are given, in $\mathcal{F}_{i-1}$, as:

$$^{\mathcal{F}_{i-1}}\mathbf{A}_{\mathcal{F}_i} = \mathbf{R}(z, \theta_i)\mathbf{D}(0, 0, r_i)\mathbf{D}(l_i, 0, 0)\mathbf{R}(x, \alpha_i). \tag{2.1}$$

where $\mathbf{R}(axis, angle)$ and $\mathbf{D}(x, y, z)$ are 4×4 homogeneous transformation matrices expressing, respectively, a rotation of an *angle* around the *axis* and the translation distance along x, y and z.

As shown in Fig. 2.1, equation (2.1) may be interpreted as a mean of transforming frame $\mathcal{F}_{i-1}$ to frame $\mathcal{F}_i$ by using the following sequential steps through joint i:

- rotate frame $\mathcal{F}_{i-1}$ around z_{i-1} by an angle θ_i, the joint angle;
- translate along z_{i-1} a distance r_i, the offset;

- translate along the rotated x_{i-1}, a distance l_i, the link length, and finally,
- rotate around x_i, the angle α_i.

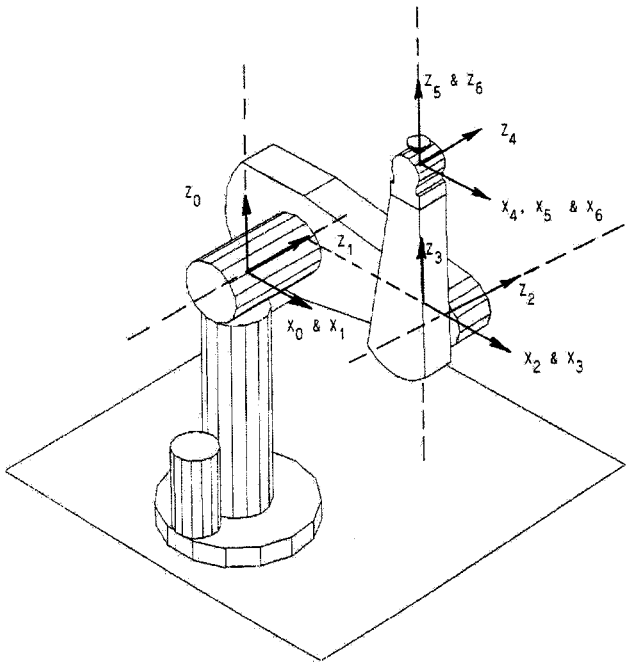


Figure 2.2 Coordinate frames of the PUMA560

As an example, Fig. 2.2 shows the PUMA 560 together with each link, joint, and coordinate frame assignment. By using the homogeneous transformation matrices ${}^{\mathcal{F}_{i-1}}\mathbf{A}_{\mathcal{F}_i}$ corresponding to each joint i as in eq.(2.1), one can write the kinematic model of any 6-DOF serial manipulator as:

$$ {}^{\mathcal{F}_0}\mathbf{A}_{\mathcal{F}_6} = {}^{\mathcal{F}_0}\mathbf{A}_{\mathcal{F}_1}{}^{\mathcal{F}_1}\mathbf{A}_{\mathcal{F}_2}{}^{\mathcal{F}_2}\mathbf{A}_{\mathcal{F}_3}{}^{\mathcal{F}_3}\mathbf{A}_{\mathcal{F}_4}{}^{\mathcal{F}_4}\mathbf{A}_{\mathcal{F}_5}{}^{\mathcal{F}_5}\mathbf{A}_{\mathcal{F}_6} \ , \tag{2.2} $$

where ${}^{\mathcal{F}_0}\mathbf{A}_{\mathcal{F}_6}$ is also a 4×4 homogeneous transformation matrix describing the position, namely $\mathbf{p}$, and orientation, namely $\mathbf{Q}$, of frame $\mathcal{F}_6$, attached to the EE, with respect to frame $\mathcal{F}_0$, attached to the base, as

$$ {}^{\mathcal{F}_0}\mathbf{A}_{\mathcal{F}_6} = \begin{bmatrix} \mathbf{Q} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} . \tag{2.3} $$

2.2.2 Levels of Kinematic Analysis

The kinematic analysis of serial manipulators comprises the study for three aspects of such mechanical systems[1]:

1. the relations between *joint positions* and *Cartesian positions* of the EE, known as *displacement analysis*;
2. the relations between the time-rates of change of the joint positions, referred to as the *joint rates*, and the twist of the EE, known as *velocity analysis*;
3. the relations between the second time-derivatives of the joint positions, referred to as the *joint accelerations*, with the time-rate of change of the twist of the EE, known as *acceleration analysis.*

For the sake of this thesis, a complete introduction of all three aspects is neither possible nor necessary. Below, we rather to introduce the *direct kinematic problem* (DKP), the *inverse kinematic problem* (IKP) at the displacement level, and the differential kinematics at the velocity level.

2.2.3 Direct and Inverse Kinematic Problems

Figure 2.3 shows the mapping between joint space and operational space at the displacement level. The DKP is the mapping from joint space to operational space, *i.e.*, the DKP consists of determining the pose of the EE, for a given manipulator in a given posture. This problem is straightforward and admits a single solution, which can be determined by simple matrix and vector multiplications. The IKP is the mapping from operational space to joint space, *i.e.*, the IKP consists of determining the posture of a given manipulator for a given pose of its EE. At the

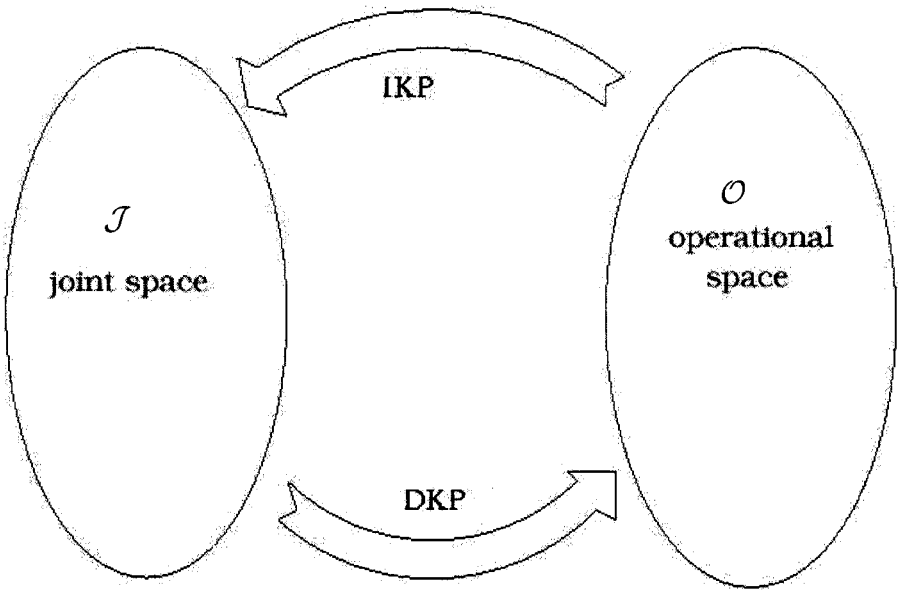


Figure 2.3 Mapping between joint space and operational space

displacement level, a point in operational space may map onto a set of points in the joint space, while a point in joint space represents a unique pose of the EE. Accounting for the dependence of position and orientation of the EE with the joint positions, the DKP can be written as the following nonlinear algebraic system, *i.e.*,

$$\mathbf{x} = \kappa(\boldsymbol{\theta}), \tag{2.4}$$

where $\boldsymbol{\theta}$ is a point in $\mathcal{J}$ and $\mathbf{x}$ the corresponding point in $\mathcal{O}$. The function $\kappa(\cdot)$ allows the computation of the operational space variables $\mathbf{x}$ from the knowledge of the joint space variables $\boldsymbol{\theta}$.

Alternatively, the IKP is also written as the following nonlinear algebraic system, *i.e.*,

$$\boldsymbol{\theta} = \eta(\mathbf{x}). \tag{2.5}$$

For general six-revolute serial manipulators, function $\eta(\cdot)$ can be a monovariate

polynomial of up to a 16th degree after applying intensive variable-elimination methods, whose roots are the solutions of the IKP[1]. The IKP is much more complex and challenging than the DKP for the following reasons[11]:

- the equations to be solved are nonlinear, and thus it is rarely possible to find a closed-form solution;
- multiple solutions may exist;
- an infinite number of solutions may exist as well, *e.g.*, in the case of a kinematically redundant manipulator;
- there might be no admissible solutions, in view of the manipulator kinematic structure. One may ask a pose of the EE outside of the reachable workspace of the manipulator.

Pieper[12] studied the solution of the IKP of general six-revolute serial manipulators. He proposed that there are up to 16 real solutions for a given Cartesian pose of the EE, without considering the mechanical joint limits. Such occurrence demands some criteria to choose the possible solutions. The existence of mechanical joint limits may eventually reduce the number of reachable solutions for the given manipulator. He also showed that a 6-DOF manipulator, with three succeeding revolute joint axes intersecting at a point, termed as *decoupled manipulator*, always has closed-form solutions.

2.2.4 Differential Kinematic Method

Whitney[13] in 1969 proposed a differential kinematic method to solve the IKP. This method uses differential relationships to solve the joint space motion for a given Cartesian space motion of the EE. The relationship between the EE velocity and

the joint velocity is represented by a linear algebraic equation. The coefficient of the linear equation is the *Jacobian matrix*, which is a nonlinear function of joint angles. Whitney named this method as *resolved motion rate* method.

At the velocity level, the relationship between $\dot{\mathbf{x}}$ and $\dot{\boldsymbol{\theta}}$ is

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}, \tag{2.6}$$

where $\dot{\mathbf{x}}$ and $\dot{\boldsymbol{\theta}}$ are, respectively, the *twist* array of the EE and the joint velocity vector, *i.e.*,

$$\dot{\mathbf{x}} \equiv \begin{bmatrix} \boldsymbol{\omega}^T & \dot{\mathbf{p}}^T \end{bmatrix}^T \in 2 \times \mathbb{R}^3. \tag{2.7}$$

$$\dot{\boldsymbol{\theta}} \equiv \begin{bmatrix} \dot{\theta}_1 & \dots & \dot{\theta}_n \end{bmatrix}^T \in \mathbb{R}^n; \tag{2.8}$$

and $\mathbf{J}$ is the Jacobian matrix of the manipulator, which is a linear mapping from the joint velocity space to the EE velocity space. Although eq.(2.6) is fully true, it can not be obtained as the time-derivative of eq.(2.4), because there is no $\mathbf{x}$ defined such as in eq.(1.2) that the time-derivative gives $\dot{\mathbf{x}}$ as defined in eq.(2.7).

The dimension size of the EE twist is always 6, and n is the size of the manipulator joint space. For the sake of this thesis, the Jacobian matrix of a serial n-axis manipulator containing only revolute pairs is considered. Hence, $\mathbf{J}$ is a function of the configuration of $\boldsymbol{\theta}$, *i.e.*, $\mathbf{J}(\boldsymbol{\theta})$.

From Fig. 2.4, the angular velocity vector of the EE, namely $\boldsymbol{\omega}$, is readily computed as

$$\boldsymbol{\omega} = \sum_{i=1}^{n} (\dot{\theta}_i \ \mathbf{e}_i), \tag{2.9}$$

where $\dot{\theta}_i$ and $\mathbf{e}_i$ are respectively defined as a joint rate and a unit vector associated with revolute axis of joint i.

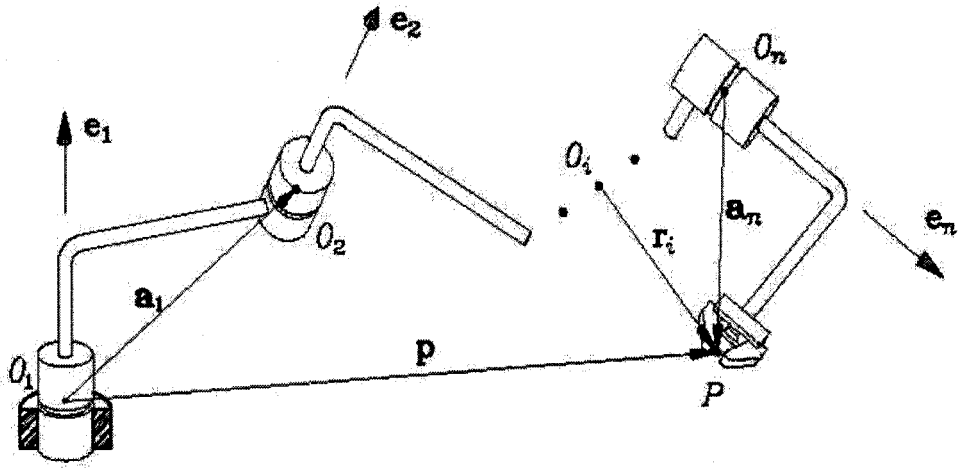


Figure 2.4 General n axis manipulator

Moreover, the translational velocity of the origin of the frame $\mathcal{F}_n$ attached to the EE, namely $\dot{\mathbf{p}}$, is readily computed as

$$\dot{\mathbf{p}} = \sum_{i=1}^{n} (\dot{\theta}_i \ \mathbf{e}_i \times \ \mathbf{r}_i), \tag{2.10}$$

where vector $\mathbf{r}_i$ is defined as the position vector of the origin of $\mathcal{F}_n$ with respect to the origin of $\mathcal{F}_i$ and expressed in $\mathcal{F}_0$, *i.e.*,

$$\mathbf{r}_i \equiv \mathbf{a}_i + \mathbf{a}_{i+1} + ... + \mathbf{a}_n, \tag{2.11}$$

and $\mathbf{a}_i$ is defined as

$$\mathbf{a}_i = \begin{bmatrix} l_i \cos(\theta_i) \\ l_i \sin(\theta_i) \\ r_i \end{bmatrix}. \tag{2.12}$$

Therefore, the Jacobian matrix $\mathbf{J}$ can be written as a $(6 \times n)$ matrix, *i.e.*,

$$\mathbf{J} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \tag{2.13}$$

where

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix} \in \mathbb{R}^{3\times n}; \tag{2.14}$$

$$\mathbf{B} \equiv \begin{bmatrix} \mathbf{e}_1 \times \mathbf{r}_1 & \mathbf{e}_2 \times \mathbf{r}_2 & \dots & \mathbf{e}_n \times \mathbf{r}_n \end{bmatrix} \in \mathbb{R}^{3\times n}. \tag{2.15}$$

The condition number of $\mathbf{J}$, namely $k(\mathbf{J})$, is defined as the ratio of the largest singular value σ_l of $\mathbf{J}$ to the smallest one, σ_s, *i.e.*,

$$k(\mathbf{J}) \equiv \frac{\sigma_l}{\sigma_s} \qquad (k(\mathbf{J}) \geq 1). \tag{2.16}$$

The $k(\mathbf{J})$ can be used as a measure of the amplification of error in differential kinematic method[14]. A $\mathbf{J}$ with a small condition number is said to be *well conditioned*, whereas one with a large condition number is said to be *ill conditioned*. These terms are used because the numerical computation of a linear equation with an ill-conditioned coefficient matrix may involve large computational errors. The worst-conditioned Jacobian is the one that is singular. The condition number of a matrix can also be understood as a measure of the relative round-off error amplification of the computed results upon solving the linear system of equations at hand with respect to the round-off error of the data [16][15].

2.3 Trajectory Planning and Programming

In order to make the description of a manipulator motion easy for a user of a robot system, the user should not be required to write down complicated functions of joint-position and take time to specify the task. Rather, we must allow the capability of specifying trajectories with simple descriptions of the desired motion, and let the system to figure out the details. For example, the user may just specify the desired goal position and orientation of the EE, let the system decide the exact path, the duration, the velocity profile, and other details.

Here, a *path* denotes the locus of points of the manipulator has to follow during the execution of the assigned motion. A *trajectory* is a path on which a time law is specified in terms of velocities and/or accelerations at each point. A path and a trajectory can be defined either in joint or operational space. Usually, the latter is preferred since it allows a natural description of the task from a user point of view.

The goal of *trajectory planning* is to generate the reference motion to the control system which ensures that the manipulator executes the planned trajectories. The user typically specifies a number of parameters to describe the desired trajectory. In principle, it can be conceived that the inputs to a *trajectory planning* algorithm are the path description, the path constraints, and the constraints imposed by manipulator dynamics, whereas the outputs are the joint or EE trajectories in terms of a time sequence of the values attained by position, velocity and acceleration.

A geometric path cannot be fully specified by the user for obvious reasons of complexity. Typically, a reduced number of parameters is specified, such as extremal points, possible intermediate points, and geometric primitive interpolating the points. Moreover, the motion time law is not typically specified at each point of the geometric path, but rather it regards the total trajectory time, the constraints on the maximum velocities and accelerations, and eventually the assignment of velocity and acceleration at particular points of interest. Base on this information, the trajectory planning algorithm generates a time sequence of variables that describe the EE position and orientation over time with respect to the imposed constraints[17]. Since the control action on the manipulator is carried out in joint space, a suitable inverse kinematic algorithm is to be used to reconstruct the time sequence of joint variables corresponding to the above sequence into the operational space.

The motion of the manipulator should be as smooth as possible, because abrupt

motion requires unlimited amounts of power to produce, which the motors can not supply. Hence, abrupt change in position, velocity, and acceleration should be avoided in trajectory planning.

There are two typical tasks for trajectory planning techniques, namely:

- pick-and-place operations (PPO), and
- continuous paths (CP).

Here, we focus our discussion on the continuous paths planning.

2.3.1 Continuous Path Planning and Tracking

Continuous-path planning appears in operations such as arc-welding, flange-cutting, and routing. In these operations, a tool is rigidly attached to the EE of a robotic manipulator, which is used to trace a continuous and smooth trajectory in a 6-dimensional operational space. Three dimensions of this space describe the spatial path expressed by the position of the EE, while the remaining three describe the orientation of the EE. For functional reasons, the orientation of the EE is given as a rotation matrix that is a prescribed smooth function of time. While most trajectory planning methods in Cartesian-coordinate level focus on the path followed by the operation point, the underlying inverse kinematic of a six-axis robotic manipulator requires the specification of the orientation of the EE as well. For some simple cases, the position and the orientation tasks are separable, hence, the planning of the two tasks can be done at the same time. However, the separation is not possible for most robotic operations, and thus both tasks must be planned concurrently.

Once a continuous trajectory in operational space is created, the EE of the robot

should track this trajectory, and hence, the joint angles of the robot have to be calculated along this continuous set of poses of the EE. In practice, the continuous trajectory is sampled at a discrete set of close-enough poses. In principle, an IKP must be solved at each sampled pose. In order to improve the calculation speed of solving the IKP, an approach which solves the IKP *iteratively* is applied. If we have the values of the joint-position $\boldsymbol{\theta}(t_k)$ at time t_k and want to find $\boldsymbol{\theta}(t_{k+1})$ at t_{k+1}, then one can use Algorithm 2.1 as proposed by Pieper (1968)[12].

Algorithm 2.1[1]

$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta}(t_k)$

1 find correction $\Delta\boldsymbol{\theta}$
if $\|\Delta\boldsymbol{\theta} \leq \epsilon\|$, then stop;
else
$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
go to 1

The procedure to find the correction $\Delta\boldsymbol{\theta}$ is based on Newton-Gauss method[18], let the following nonlinear algebraic system of seven equations in six unknowns as

$$\mathbf{f}(\boldsymbol{\theta}) = \mathbf{s}_d, \tag{2.17}$$

where $\mathbf{s}_d$ is the 7-dimensional *prescribed-pose array* as

$$\mathbf{s}_d \equiv \begin{bmatrix} \mathbf{q} \\ q_o \\ \mathbf{p} \end{bmatrix}_d, \tag{2.18}$$

with $\mathbf{q}$ and q_0 defined, in turn, as a 3-dimensional vector invariant of the rotation $\mathbf{Q}$ and its corresponding scalar, respectively. Vector $\mathbf{p}$ is the position vector of the operation point. In this context, the direct kinematics is given as the 7-dimensional

vector $\mathbf{f}$, *i.e.*,

$$\mathbf{f}(\boldsymbol{\theta}) \equiv \begin{bmatrix} \mathbf{f}_v(\boldsymbol{\theta}) \\ f_o(\boldsymbol{\theta}) \\ \mathbf{f}_p(\boldsymbol{\theta}) \end{bmatrix}. \tag{2.19}$$

Upon application of the Newton-Gauss method to find a solution of eq.(2.17), we assume that we have an initial guess $\boldsymbol{\theta}^0$, and based on this value, we generate a sequence $\boldsymbol{\theta}^1$, ..., $\boldsymbol{\theta}^i$, $\boldsymbol{\theta}^{i+1}$,..., until either a convergence or an abortion criterion is met. This sequence is generated in the form

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i + \Delta\boldsymbol{\theta}^i \tag{2.20}$$

with $\Delta\boldsymbol{\theta}^i$ calculated from

$$\boldsymbol{\Phi}(\boldsymbol{\theta}_i)\Delta\boldsymbol{\theta}^i = -\mathbf{f}(\boldsymbol{\theta}^i) + \mathbf{s}_d, \tag{2.21}$$

where $\boldsymbol{\Phi}$ is defined as

$$\boldsymbol{\Phi} \equiv \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}. \tag{2.22}$$

In 1968, Pieper[12] proposed a numerical path-tracking scheme, in which the computation of the correction $\Delta\boldsymbol{\theta}$ involves only linear-equation solving, *i.e.*,

$$\Delta\boldsymbol{\theta}^i = \boldsymbol{\Phi}^{-1}(\boldsymbol{\theta}_i)(-\mathbf{f}(\boldsymbol{\theta}^i) + \mathbf{s}_d), \tag{2.23}$$

Alternatively, Whitney(1972)[19] avoids the use of this differential form of the Jacobian matrix and proposed to use a vectorial form of the latter as in eq.(2.13), and hence, we have

$$\mathbf{J}\Delta\boldsymbol{\theta} = \Delta\mathbf{x}, \tag{2.24}$$

with $\Delta\mathbf{x}$ defined as

$$\Delta\mathbf{x} \equiv \begin{bmatrix} \mathbf{Q}_k \mathbf{vect}(\mathbf{Q}_k^T \mathbf{Q}_d) \\ \Delta\mathbf{p} \end{bmatrix}, \tag{2.25}$$

where $\mathbf{Q}_k$ represents the rotation matrix from base frame to EE frame at time t_k, $\mathbf{Q}_d$ represents the desired rotation matrix at time t_{k+1}, and they have a relation as

$$\mathbf{Q}_d = \mathbf{Q}_k \Delta\mathbf{Q}. \tag{2.26}$$

Vector $\Delta\mathbf{p}$ defined as the difference between the desired value $\mathbf{p}_d$ of the position vector of the operation point and its actual value $\mathbf{p}_k$. The relation among $\mathbf{Q}_d$, $\mathbf{p}_d$, $\mathbf{Q}_k$, $\mathbf{p}_k$, and $\Delta\mathbf{Q}$, $\Delta\mathbf{p}$ are shown in Fig. 2.5. Thus, we have Algorithm 2.2. Since any norm can be used to calculate the vector norm $\|\Delta\boldsymbol{\theta}\|$, we can choose the norm that is fastest to compute. Function $\mathbf{vect}(\Delta\mathbf{Q})$ represents the axial vector of a 3×3 rotation matrix $\Delta\mathbf{Q}$, and is calculated as

$$\mathbf{vect}(\Delta\mathbf{Q}) \equiv \frac{1}{2} \begin{bmatrix} q_{32} - q_{23} \\ q_{13} - q_{31} \\ q_{21} - q_{12} \end{bmatrix}. \tag{2.27}$$

2.3.2 Robot Programming

In general, there are five different methods to enter a robot program into the robot controller[20]:

1. Manual programming;
2. Teach-pendant programming;
3. Walk-through programming;

Algorithm 2.2[1]

1 $\Delta\mathbf{Q} \longleftarrow \mathbf{Q}_k^T\mathbf{Q}_d$

$\Delta\mathbf{p} \longleftarrow \mathbf{p}_k - \mathbf{p}_d$

$$\Delta\mathbf{x} \longleftarrow \begin{bmatrix} \mathbf{Q}_k\mathbf{vect}(\Delta\mathbf{Q}) \\ \Delta\mathbf{p} \end{bmatrix}$$

2 $\Delta\boldsymbol{\theta} \longleftarrow \mathbf{J}^{-1}\Delta\mathbf{x}$

if $\|\Delta\boldsymbol{\theta}\| \leq \epsilon$, then stop;

else

$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

$\mathbf{Q}_k \longleftarrow \mathbf{Q}_k\Delta\mathbf{Q}$

$\mathbf{p}_k \longleftarrow \mathbf{p}(\boldsymbol{\theta})$

go to 1

4. Computer terminal on-line programming;

5. Off-line programming;

Manual programming is regulated by a sequencing device (stepping switch, cam) and has limited flexibility. *Teach pendant programming* involves the use of a teach pendant for controlling the movements of the manipulator joints. This method is easy to learn and suitable for programming point-to-point tasks in industry. *Walk-through programming* requires the operator to physically grasp the end-effector (EE) and manually move it through the motion sequence, recording the path into memory. However, these three methods cannot enter program into controller while the robot is off-line (away from the robot workcell).

Computer terminal programming uses textual language commands which are written in English-like statements to perform the programming. It can be done on-line or off-line.

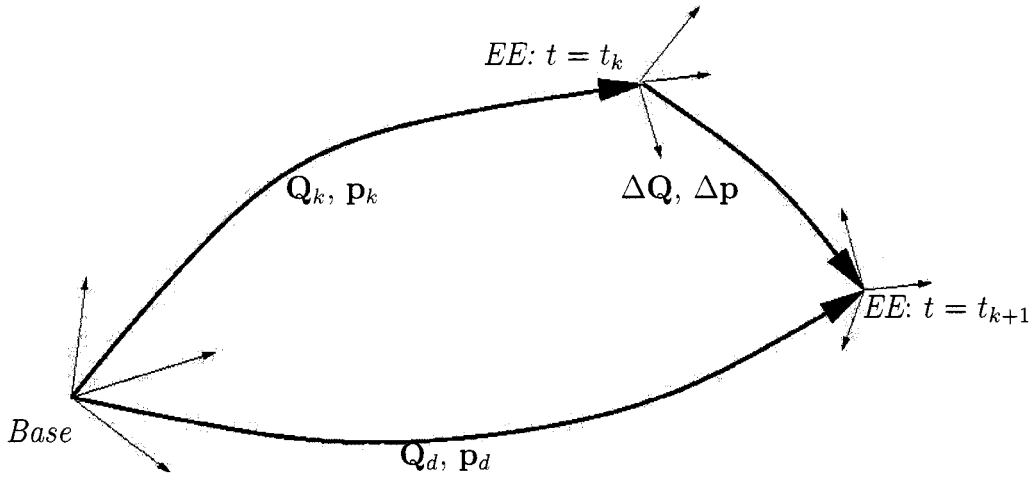


Figure 2.5 Relation between the desired posture and the current postures

Off-line programming prepare the robot programming at a remote computer terminal and download it to the robot controller for execution. Off-line programming systems, generally including a computer graphic interface, which allows the robots to be programmed without access to the robot itself during the programming. Off-line programming have at least the following advantages:

1. reducing down-time caused by robot reprogramming;
2. avoiding the risk of damage to real robot by checking the motion on graphic simulator in advance;
3. it becomes possible that existing computer-aider design (CAD) and computer-aider manufacturing (CAM) information are incorporate into the control functions.

Concerning these advantages, the off-line programming has already attracted a lot of attention of researchers. For example, Monsberger[21] developed a graphical simulator, Lee[22] developed a system which not only included a graphical simulation

model, but also provided a language translation software package that could automatically generate the necessary programming commands for the manipulator. However, Lee's system is used on 5-axis robot, and does not consider redundant tasks.

2.4 Kinematic Redundancy

For the sake of the following discussion, let us define $\mathbf{t} \equiv \dot{\mathbf{x}}$, then we can rewrite eq.(2.6) as

$$\mathbf{t} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}. \tag{2.28}$$

If $\mathbf{J}(\boldsymbol{\theta})$ is square and nonsingular, $\dot{\boldsymbol{\theta}}$ can be computed as

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\mathbf{t}. \tag{2.29}$$

If $\mathbf{J}(\boldsymbol{\theta})$ has fewer rows than columns or has a rank less than the number of rows, it is an under-determined case and there will be an infinite number of solutions.

Suppose $\dot{\boldsymbol{\theta}}_1$ and $\dot{\boldsymbol{\theta}}_2$ are two distinct solutions of an under-determined case for a given $\mathbf{t}$, one can write

$$\dot{\boldsymbol{\theta}}_e = \dot{\boldsymbol{\theta}}_1 - \dot{\boldsymbol{\theta}}_2 \neq \mathbf{0}, \tag{2.30}$$

and hence substituting eq.(2.30) into eq.(2.28), we have

$$\mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_e = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_1 - \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_2 = \mathbf{0}. \tag{2.31}$$

Equations (2.30) and (2.31) imply that the difference of two solutions, *i.e.* $\dot{\boldsymbol{\theta}}_e$, is always mapped onto the zero vectors in velocity space of the EE.

The relationship in eq.(2.28) can be characterized in terms of the *range* and *null*

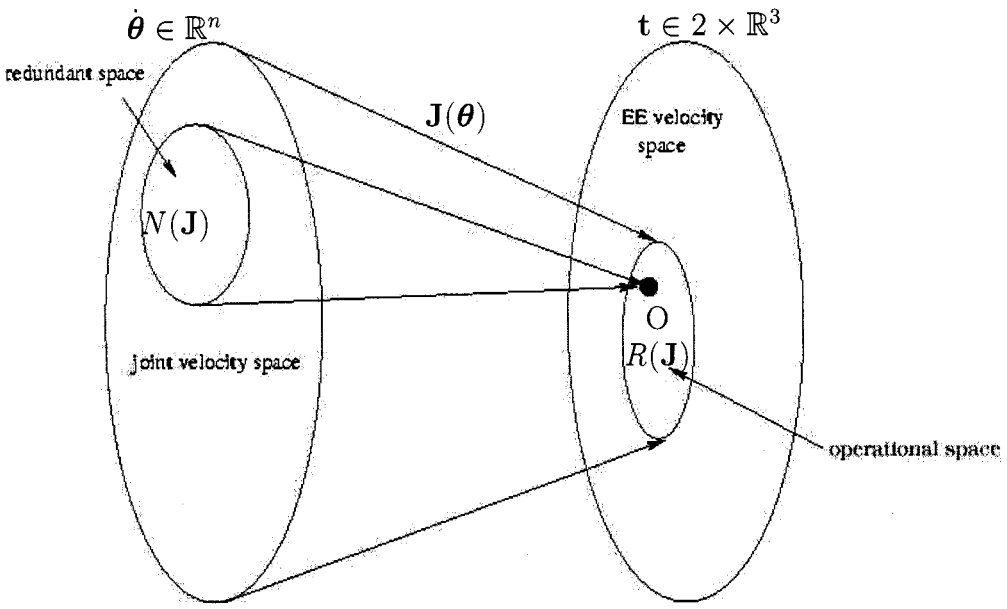


Figure 2.6 Mapping between the redundant space and the operational space at the velocity level

spaces of the mapping.

The *range* of $\mathbf{J}$, namely $R(\mathbf{J})$, is the operational subspace of the EE velocities space $2 \times \mathbb{R}^3$ that can be generated by the joint velocities $\dot{\boldsymbol{\theta}}$, of the given manipulator posture $\boldsymbol{\theta}$. The range space of $\mathbf{J}$ is called the *operational space* of the EE.,

The *null* subspace of $\mathbf{J}$, namely $N(\mathbf{J})$, is the joint space $\mathbb{R}^n$ that do not produce any EE velocity $\mathbf{t}$ of a given manipulator posture $\boldsymbol{\theta}$. The null space of $\mathbf{J}$ is called the *redundant space.*

The range space and null space of $\mathbf{J}$ should satisfy the following equation:

$$dim(R(\mathbf{J})) + dim(N(\mathbf{J})) = n. \tag{2.32}$$

In the context of differential kinematic, the Jacobian matrix describes the mapping

from the joint velocity space to the EE velocity space, and has to be regarded as a constant matrix, since the instantaneous velocity mapping is of interest for a given posture. The mapping is schematically illustrated in Fig. 2.6[11].

2.5 Intrinsic Redundancy-Resolution Schemes

2.5.1 Classification of Intrinsic Redundancy-Resolution Schemes

Redundant manipulators attracted many researchers in recent years, because their extra DOF allow for more sophisticated motions than their non-redundant counterparts. Hence, the redundant manipulators have widely been used on avoidance tasks for obstacles, joint-limits and singularities. Many research works focused on the theoretical aspects of the RR schemes. For example, Baillieul[23][24] used redundancy for obstacle avoidance by using an extended Jacobian technique; Klein [25] solved redundancy for maximum dexterity of a manipulator. However, most of the reported works have based only on simulations, while only a few implementations on real robots have been reported as Honegger and Codourey[26].

Chahbaz(1994)[27] suggested a joint-rate compensation method to minimize the implementation errors of redundant manipulator and verified it for a planar redundant manipulator. This method directly separate the manipulator joints into two sub-groups: the first includes the joints with a given joint-rate, and the joints of the second group move to follow the task trajectory. Hence, there is equation as

$$\mathbf{t} = \mathbf{J}_v \dot{\boldsymbol{\theta}}_v + \mathbf{J}_a \dot{\boldsymbol{\theta}}_a, \tag{2.33}$$

where $\mathbf{J}_v$ and $\dot{\boldsymbol{\theta}}_v$ are the Jacobian component and the joint-rate related to the joints of the first group, respectively. Similarly, $\mathbf{J}_a$ and $\dot{\boldsymbol{\theta}}_a$ are related to the joints of the

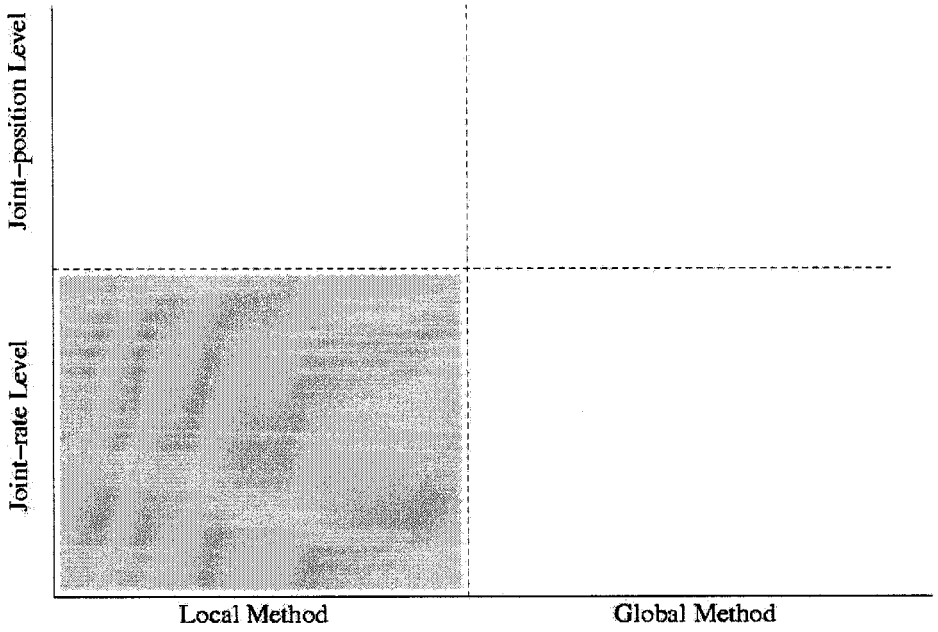


Figure 2.7 Classification of RR schemes

second group. As $\dot{\boldsymbol{\theta}}_v$ are given to be constant or with constant acceleration, the joint-rate of the second group can be solved to meet the task trajectory,

$$\dot{\boldsymbol{\theta}}_a = \mathbf{J}_a^{-1}[\mathbf{t} - \mathbf{J}_v\dot{\boldsymbol{\theta}}_v]. \tag{2.34}$$

Clearly, this method is limited to the cases where the separation of eq.(2.33) is possible. Unfortunately, these separation are only rarely possible and can not be used in general.

The RR schemes can be classified into local methods and global methods. The global optimal control methods need all the required data before the movement is realized. The local optimal control methods give a solution for every instant and can also use the available sensor data. Thus, the locally optimal control approach is suitable for real-time applications, such as sensor-based obstacle avoidance strategies. The globally optimal control approach is limited only on the off-line trajectory

planning for tasks requiring strict optimality, such as obstacle avoidance in a complicated but time invariant workspace and energy minimization[6]. Therefore, these two frameworks should be properly used, depending on the situation. Most of the RR schemes use local methods since their performance index depends on the instantaneous robot posture, this thesis also use a local method to realize optimization. Hence, this literature review only cover the local methods.

The RR problem has been solved at both joint-position level[28][29] and joint-rate level[24][30] using different methods. At the joint-position level, the RR problem is nonlinear, whereas it is linear at the joint-rate level. Figure 2.7 shows the classification of RR schemes discussed above. The shaded part is our research domain of interest, *i.e.*, RR scheme in local method of joint-rate level.

Now a day, most of the RR researchers worked on the joint-rate level, and used the Moore-Penrose Generalized Inverse[1] (GI) or the Weighted Generalized Inverse (WGI) of the Jacobian matrix. Therefore, the methods using GI and WGI are investigated further in the following sections.

---

[1][31] Given an $m \times n$ matrix $\mathbf{B}$, the Moore-Penrose generalized matrix inverse is a unique $n \times m$ matrix pseudoinverse $\mathbf{B}^\dagger$. The Moore-Penrose inverse satisfies

$$\mathbf{B}\mathbf{B}^\dagger\mathbf{B} = \mathbf{B}$$

$$\mathbf{B}^\dagger\mathbf{B}\mathbf{B}^\dagger = \mathbf{B}^\dagger$$

$$(\mathbf{B}\mathbf{B}^\dagger)^T = \mathbf{B}\mathbf{B}^\dagger$$

$$(\mathbf{B}^\dagger\mathbf{B})^T = \mathbf{B}^\dagger\mathbf{B}.$$

It is also true that

$$\mathbf{z} = \mathbf{B}^\dagger\mathbf{c}$$

is the shortest length solution to the problem

$$\mathbf{B}\mathbf{z} = \mathbf{c}.$$

2.5.2 Schemes Using the Generalized Inverse

The Moore-Penrose pseudo-inverse can be used to find the joint-rate vector that has the smallest Euclidean norm, usually called the minimum-norm solution computed as

$$\mathbf{J}^{\dagger} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}, \tag{2.35}$$

and hence eq.(2.28) becomes

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\dagger}\mathbf{t}. \tag{2.36}$$

Klein and Huang (1983)[32] showed that such a solution may lead to noncyclic motions in joint space. At the cost of giving up the minimum-norm solution, an homogeneous component can be added to eq.(2.36) in order to optimize a secondary task into an additional criterion. Thus, this non-minimum-norm general inverse kinematic solution can be written as:

$$\dot{\boldsymbol{\theta}} = \underbrace{(\mathbf{J}^{\dagger})\mathbf{t}}_{\text{minimum-norm solution}} + \underbrace{(\mathbf{1} - \mathbf{J}^{\dagger}\mathbf{J})\mathbf{h}}_{\text{homogeneous solution}}, \tag{2.37}$$

where the first part of eq.(2.37) is the minimum-norm solution, known as the base solution, and the second part is an arbitrary vector from the null space of the Jacobian. Equation (2.37) is used widely by many researchers such as Angeles(1998)[33], and Siciliano(1992)[30] in order to solve redundant tasks. Vector $\mathbf{h}$ of eq.(2.37) is an optimized performance criterion. Different selection of $\mathbf{h}$ will give different performances.

Liégeois[34] developed the *Gradient projection method* (GPM), which minimizes a position-dependent scalar performance index $p(\boldsymbol{\theta})$ and takes its gradient as the vector $\mathbf{h}$.

$$\mathbf{h} = w\frac{\partial p}{\partial \boldsymbol{\theta}} = w\left[\begin{array}{cccc} \frac{\partial p}{\partial \theta_1} & \frac{\partial p}{\partial \theta_2} & \cdots & \frac{\partial p}{\partial \theta_n} \end{array}\right]^{T}, \tag{2.38}$$

where w is a positive scalar coefficient. Liégeois introduced a p that helps joint-limit avoidance, in the form

$$p = \frac{1}{n}\sum_{i=1}^{n}(\frac{\theta_i - \theta_i^{mid}}{\theta_i^{mid} - \theta_i^{max}})^2, \tag{2.39}$$

where $\theta_i^{mid} = (\theta_i^{min} + \theta_i^{max})/2$, θ_i^{min} and θ_i^{max} are lower and upper joint limits, respectively.

Other researchers developed different performance index related to varied applications of RR schemes.

Yoshikawa(1984)[35] suggested a scaler value μ to measure the *manipulability*, namely,

$$\mu = \sqrt{det(\mathbf{J}\mathbf{J}^T)}, \tag{2.40}$$

and used μ as performance index p to avoid singularities. Yoshikawa also introduced a performance index for obstacle avoidance, namely,

$$p = \frac{(\boldsymbol{\theta} - \boldsymbol{\theta}_r)^T\mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_r)}{2}, \tag{2.41}$$

where $\mathbf{H}$ is a diagonal matrix with constant entries greater than zero and $\boldsymbol{\theta}_r$ is a given arm posture.

Yoshikawa's manipulability measure has been used extensively. However, the determinant cannot be a measure of how close a matrix is to singularity, as pointed out by Golub and Van Loan (1989)[16]. Therefore, Kosuge and Furuta (1985)[36] suggested to take the reciprocal (in order to have a number between 0 and 1) of the condition number of the Jacobian matrix as a *controllability measure.*

Hanafusa, Yoshikawa and Nakamura (1981)[37] decomposed a task into several subtasks and assigned them priorities, then each task is solved with the order of priorities.

Damped least-square methods introduce a damping factor ρ to the solution, namely,

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_{\tau}^{\dagger}\mathbf{t}, \tag{2.42}$$

where

$$\mathbf{J}_{\tau}^{\dagger} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T} + \rho\mathbf{1})^{-1}. \tag{2.43}$$

Obviously, the damping factor affects the manipulator performance. Nakamura and Hanafusa (1986)[38] suggested an automatic adjustment of the damping factor, which affects the manipulator performance in the neighborhood of singularities, in the form as

$$\rho = \begin{cases} \rho_0(1 - \mu/\mu_0)^2, & \text{if } \mu < \mu_0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.44}$$

where μ was manipulability as defined in eq.(2.40), ρ_0 is a constant which represents the scale factor at a singularity and μ_0 is a threshold which represents the boundary neighborhood of singular points.

Instead of determining the threshold μ_0 and testing whether $\mu < \mu_0$, Kelmar and Khosla (1990)[39] suggested to use the ratio of μ between two iterations. The ratio will drop dramatically as the manipulator approaches a singularity. Therefore, their damping factor is

$$\rho = \begin{cases} \rho_0(1 - \mu^{i+1}/\mu^{i})^2, & \text{if } \mu^{i+1}/\mu^{i} < k; \\ 0, & \text{otherwise.} \end{cases} \tag{2.45}$$

where k is a proper threshold determined experimentally.

2.5.3 Schemes Using the Weighted Generalized Inverse

The WGI solution to eq.(2.28) is

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_w^{\dagger}\mathbf{t}, \tag{2.46}$$

where

$$\mathbf{J}_w^{\dagger} = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1}, \tag{2.47}$$

and $\mathbf{W}$ is a positive-definite weighting matrix. Whitney (1969)[13] applied priorities through the weighted matrix. Park, Chung and Youm (1996)[40] used the WGI and introduced the weight not only in the generalized inverse, but also in the Jacobian matrix and in the joint velocity, which they called *weighted Jacobian* and *weighted joint velocity*, respectively. Hence, they solved the equation

$$\mathbf{J}_w\dot{\boldsymbol{\theta}}_w = \mathbf{t}, \tag{2.48}$$

where $\mathbf{J}_w$ and $\dot{\boldsymbol{\theta}}_w$ are the weighted Jacobian and weighted joint velocity, respectively.

Chan and Dubey (1993)[41] compared WGI with GPM in the case of joint limits avoidance problem, and found WGI reached a more accessible joint trajectory than GPM.

2.5.4 Scheme Using Householder Reflection

Arenson, Angeles and Slutski(1998)[33] proposed to use *Householder reflection* in RR scheme. For the sake of brevity, we name it as AA householder reflection algorithm.

The algorithm is developed as follows. At first, equation (2.37) can be rewritten as:

$$\dot{\boldsymbol{\theta}} = \mathbf{k} + \mathbf{h}, \tag{2.49}$$

where $\mathbf{k}$ is shown as:

$$\mathbf{k} = \mathbf{J}^{\dagger}(\mathbf{t} - \mathbf{J}\mathbf{h}). \tag{2.50}$$

Equation (2.50) can be rewritten as:

$$\mathbf{J}\mathbf{k} = \mathbf{t} - \mathbf{J}\mathbf{h}. \tag{2.51}$$

For solving eq.(2.51), Householder reflection is used for the Jacobian matrix $\mathbf{J}$. The matrix $\mathbf{H}$ and $\mathbf{U}$ are reached, and they have a relation with $\mathbf{J}$ in the form

$$\mathbf{H}\mathbf{J}^{T} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0} \end{bmatrix}, \tag{2.52}$$

where $\mathbf{U}$ is a $(o \times o)$ upper-triangular matrix, $\mathbf{H}$ is an orthogonal matrix $(n \times n)$, and $n > o$ for intrinsic redundancy. Hence, there is

$$\mathbf{H}^{T}\mathbf{H} = \mathbf{1}, \tag{2.53}$$

Rewriting eq.(2.51) to,

$$\mathbf{J}\mathbf{H}^{T}\mathbf{H}\mathbf{k} = \mathbf{t} - \mathbf{J}\mathbf{h}. \tag{2.54}$$

Because of:

$$\mathbf{J}\mathbf{H}^{T} = (\mathbf{H}\mathbf{J}^{T})^{T} = \begin{bmatrix} \mathbf{U}^{T} & \mathbf{0}^{T} \end{bmatrix}, \tag{2.55}$$

equation (2.54) is equal to

$$\begin{bmatrix} \mathbf{U}^{T} & \mathbf{0}^{T} \end{bmatrix} \mathbf{H}\ \mathbf{k} = \mathbf{t} - \mathbf{J}\mathbf{h}. \tag{2.56}$$

Then, $\mathbf{k}$ can be reached as follows:

$$\mathbf{k} = \mathbf{H}^T \begin{bmatrix} (\mathbf{U}^T)^{-1}(\mathbf{t} - \mathbf{Jh}) \\ \mathbf{0} \end{bmatrix}. \tag{2.57}$$

Substituting eq.(2.57) into eq.(2.49) and yields

$$\dot{\boldsymbol{\theta}} = \mathbf{H}^T \begin{bmatrix} (\mathbf{U}^T)^{-1}(\mathbf{t} - \mathbf{Jh}) \\ \mathbf{0} \end{bmatrix} + \mathbf{h}. \tag{2.58}$$

If we define:

$$\mathbf{y} \equiv \begin{bmatrix} (\mathbf{U}^T)^{-1}(\mathbf{t} - \mathbf{Jh}) \\ \mathbf{0} \end{bmatrix}, \tag{2.59}$$

Equation (2.58) can be rewritten as:

$$\dot{\boldsymbol{\theta}} = \mathbf{H}^T\mathbf{y} + \mathbf{h}. \tag{2.60}$$

The AA algorithm avoid the direct calculation of the generalized inverse of the Jacobian matrix, hence the squaring of the condition number of the Jacobian matrix is avoided and the round-off error of the algorithm is not amplified. It is an interesting solution for the calculation of ill-condition postures because here the Jacobian matrix $\mathbf{J}$ has a very high condition number.

2.6 Functional Redundancy-Resolution Schemes

In the case of functional redundant tasks, $\mathbf{J}$ in eq.(2.28) is square, and the dimension of null-subspace of $\mathbf{J}$ is zero, so the intrinsic RR schemes working on null-subspace of $\mathbf{J}$ can not be directly used. In order to change eq.(2.28) to an under-determined system, there are two possibilities. One is augmenting the dimension of joint-rate $\dot{\theta}$, another one is reducing the dimension of twist $\mathbf{t}$. Corresponding to the two

possibilities, the functional RR schemes can be classified into two groups, *i.e.*, *augmented approach* and *reduced approach*, as shown in Fig. 2.8. Virtual joint method, proposed by Baron[42], follows the augmented approach. Reduced approach includes *elimination method* and twist decomposition method. It is possible that other methods exist but we have not found them yet.

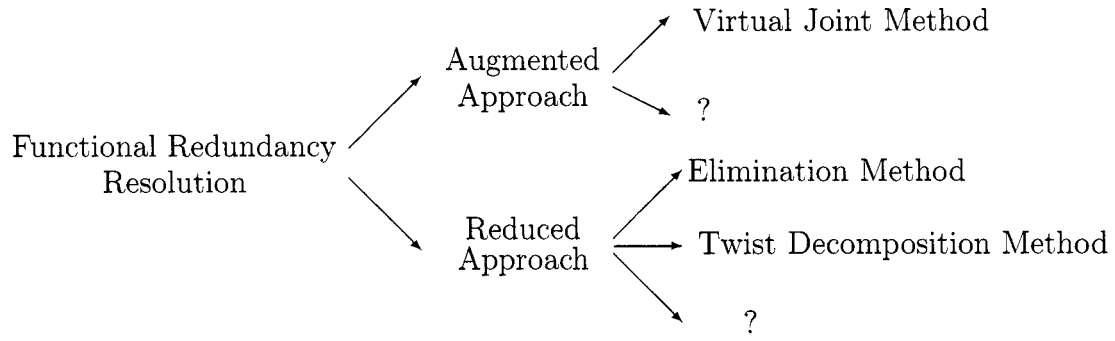


Figure 2.8 Classification of functional redundancy-resolution schemes

2.6.1 Elimination Method

In some specific cases, the redundant velocity in operational space can be identified and directly eliminated, and hence the functionally-redundant task is transformed to an intrinsically-redundant task. This method is called as elimination method.

Baron[43] proposed to use an elimination method to optimize the surfacing operation implemented by 5-axes CNC milling machine. In the case of surface milling, the orientation of EE is irrelevant to the task, and the angular velocities can be eliminated from twist $\mathbf{t}$. Thus, equation(2.28) can be rewritten as

$$\dot{\mathbf{p}} = \mathbf{B}\dot{\boldsymbol{\theta}}, \tag{2.61}$$

where $\mathbf{B}$ is lower part of $\mathbf{J}$ as defined in eq.(2.15), *i.e.*, a 3×5 matrix, while $\dot{\mathbf{p}}$ and $\dot{\boldsymbol{\theta}}$ are 3 and 5−dimensional vectors, respectively. Hence, the well known

non-minimum-norm solutions to eq.(2.61) can be written as

$$\dot{\boldsymbol{\theta}} = (\mathbf{B}^{\dagger})\dot{\mathbf{p}} + (\mathbf{1} - \mathbf{B}^{\dagger}\mathbf{B})\mathbf{h}, \tag{2.62}$$

with $\mathbf{h}$ being a secondary task and $\mathbf{B}^{\dagger}$ the right generalized inverse of $\mathbf{B}$, *i.e.*, $\mathbf{B}^{\dagger} = \mathbf{B}^{T}(\mathbf{B}\mathbf{B}^{T})^{-1}$.

However, the redundant velocity in operational space is not always constant for general tasks, and hence, it is impossible to identify the redundant velocity in operational space as above. Therefore, the elimination method can not always be applied to solve functional redundancy.

2.6.2 Virtual Joint Method

Virtual joint method changes a functionally-redundant manipulator to an intrinsically-redundant manipulator by adding a virtual joint. Although the virtual joint does not exist, it is added into the joint vector in order to obtain an under-determined linear algebraic system with at least one DOF of redundancy.

In the case of arc-welding, a virtual joint around the symmetric axis of the electrode is added. The Jacobian of the robot augmented by the virtual joint-rate $\dot{\theta}_{n+1}$, namely $\mathbf{J}_v$, maps the augmented joints-rate $\dot{\boldsymbol{\theta}}_v$ into the twist $\mathbf{t}$ of the EE as

$$\mathbf{t} = \mathbf{J}_v\dot{\boldsymbol{\theta}}_v, \tag{2.63}$$

where $\dot{\boldsymbol{\theta}}_v$ is defined as

$$\dot{\boldsymbol{\theta}}_v \equiv \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\theta}_{n+1} \end{bmatrix}. \tag{2.64}$$

The control of this virtual $(n+1)$-DOF robot along a fully constrained n-DOF task

with $\mathbf{t} = \mathbf{J}_v\dot{\boldsymbol{\theta}}_v$ is equivalent to the control of the original n-DOF robot along the $(n-1)$-DOF task with $\mathbf{t} = \mathbf{J}\dot{\boldsymbol{\theta}}$. In both cases, there is one DOF redundancy.

Hence, based on eq.(2.37), the non-minimum-norm solutions to eq.(2.63) can be written as

$$\dot{\boldsymbol{\theta}}_v = \mathbf{J}_v^{\dagger}\mathbf{t} + (\ \mathbf{1} - \ \mathbf{J}_v^{\dagger}\mathbf{J}_v)\mathbf{h}, \tag{2.65}$$

where $\mathbf{1}$ denotes the $(n+1) \times (n+1)$ identity matrix, $\mathbf{h}$ is the secondary task and $\mathbf{J}_v^{\dagger}$ is the right generalized inverse of $\mathbf{J}_v$, *i.e.*,

$$\mathbf{J}_v^{\dagger} = \mathbf{J}_v^T(\mathbf{J}_v\mathbf{J}_v^T)^{-1} \tag{2.66}$$

2.7 Conclusion

The related fundamental knowledge and background of our research is reviewed in this chapter, including the kinematics of serial manipulators, continuous path planning, robot programming, and some early works on RR.

Based on this literature review, we notice that most of RR researchers focus on the intrinsic redundancy, while omitting the existence of functional redundancy. However, we can not directly use the RR schemes developed for intrinsic redundancy to solve functional redundancy. Therefore, an appropriate RR scheme specific to functionally redundant manipulator is needed, and will be developed in the next two chapters.

CHAPITRE 3

TWIST DECOMPOSITION METHOD

3.1 Introduction

Most of the complicated tasks given to robots with many DOF could be decomposed into several subtasks with some order of priority (Yoshikawa[35]). Determining the suitable subtask depends on the requirement of the task itself. In this chapter, after introducing the orthogonal decomposition of vectors and twists, we formulate the inverse kinematics of functionally-redundant manipulators by projecting the velocity relationship onto the instantaneous-task subspace, thereby producing the so-called twist decomposition method. The paper developing twist decomposition method has been accepted by the 2005 CCToMM Symposium.

3.2 Orthogonal-Decomposition of Vectors

Let us decompose any vector $(\cdot)$ of $\mathbb{R}^3$ into two orthogonal parts, $[\cdot]_M$, the component lying on the subspace, $\mathcal{M}$, and $[\cdot]_{M^\perp}$, the component lying in the orthogonal subspace, $\mathcal{M}^\perp$, using the *projector* $\mathbf{M}$ and an *orthogonal complement* of $\mathbf{M}$, namely $\mathbf{M}^\perp$, as follows:

$$(\cdot) = [\cdot]_M + [\cdot]_{M^\perp} = \mathbf{M}(\cdot) + \mathbf{M}^\perp(\cdot) = (\mathbf{M} + \mathbf{M}^\perp)(\cdot) \tag{3.1}$$

It is apparent from eq.(3.1), that $\mathbf{M}$ and $\mathbf{M}^\perp$ are related by $\mathbf{M} + \mathbf{M}^\perp = \mathbf{1}$ and $\mathbf{M}\mathbf{M}^\perp = \mathbf{O}$, where $\mathbf{1}$ and $\mathbf{O}$ are the 3×3 identity and zero matrices, respectively.

The orthogonal complement of $\mathbf{M}$ thus defined, $\mathbf{M}^{\perp}$, is therefore unique, and hence, both $\mathbf{M}$ and $\mathbf{M}^{\perp}$ are projectors that verify the following properties:

- Symmetry: $[\mathbf{M}]^T = \mathbf{M}, \quad [\mathbf{M}^{\perp}]^T = \mathbf{M}^{\perp}$
- Idempotency: $[\mathbf{M}]^2 = \mathbf{M}, \quad [\mathbf{M}^{\perp}]^2 = \mathbf{M}^{\perp}$
- Rank-complementarity: $\text{rank}(\mathbf{M}) + \text{rank}(\mathbf{M}^{\perp}) = 3$
- Subspace-complementarity: $\mathcal{M} \oplus \mathcal{M}^{\perp} = \mathbb{R}^3$

The projector $\mathbf{M}$ projects vectors of $\mathbb{R}^3$ onto the subspace $\mathcal{M}$, while the orthogonal projector $\mathbf{M}^{\perp}$ projects those vectors onto the orthogonal subspace $\mathcal{M}^{\perp}$. These projectors are given for the four possible dimensions i of subspaces of $\mathbb{R}^3$ as:

$$\mathbf{M}_i = \begin{cases} \mathbf{1} \\ \mathbf{P} \\ \mathbf{L} \\ \mathbf{O} \end{cases}, \quad \mathbf{M}_i^{\perp} = \begin{cases} \mathbf{O} & i = 3 \quad \Rightarrow \text{3-D task} \\ \mathbf{L} & i = 2 \quad \Rightarrow \text{2-D task} \\ \mathbf{P} & i = 1 \quad \Rightarrow \text{1-D task} \\ \mathbf{1} & i = 0 \quad \Rightarrow \text{0-D task} \end{cases}, \tag{3.2}$$

where the plane and line projectors, $\mathbf{P}$ and $\mathbf{L}$, respectively, are defined as:

$$\mathbf{P} \equiv \mathbf{1} - \mathbf{L}, \quad \mathbf{L} \equiv \mathbf{e}\mathbf{e}^T, \tag{3.3}$$

in which $\mathbf{e}$ is a unit vector along the line $\mathcal{L}$ and normal to the plane $\mathcal{P}$. The null-projector $\mathbf{O}$ is the 3×3 zero matrix that projects any vector of $\mathbb{R}^3$ onto the null-subspace, while the identity-projector $\mathbf{1}$ is the 3×3 identity matrix that projects any vector of $\mathbb{R}^3$ onto itself.

3.3 Orthogonal-Decomposition of Twists

In general, the twist of a rigid body can be regarded as a 6-dimensional array[1] defining completely the velocity field of a rigid body, and it comprises the three components of the angular velocity and the three components of the translation velocity of the body.

Any twist array $(\cdot)$ of $2 \times \mathbb{R}^3$ can also be decomposed into two orthogonal parts, $[\cdot]_{\mathcal{T}}$, the component lying in the task subspace, $\mathcal{T}$, and $[\cdot]_{\mathcal{T}^\perp}$, the component lying in the orthogonal task subspace (also designated as the redundant subspace), $\mathcal{T}^\perp$, using the *twist projector* $\mathbf{T}$ and an *orthogonal complement* of $\mathbf{T}$, namely $\mathbf{T}^\perp$, as follows:

$$(\cdot) = [\cdot]_{\mathcal{T}} + [\cdot]_{\mathcal{T}^\perp} = \mathbf{T}(\cdot) + \mathbf{T}^\perp(\cdot) = (\mathbf{T} + \mathbf{T}^\perp)(\cdot) \tag{3.4}$$

It is apparent from eq.(3.4), that $\mathbf{T}$ and $\mathbf{T}^\perp$ are projectors of twists that must verify all the properties of projectors of section 3.2. However, twists are not vectors of $\mathbb{R}^6$, and hence, projectors of twists cannot be defined as in eqs.(3.2) and (3.3), *e.g.*,

$$\mathbf{T} \neq \mathbf{t}\mathbf{t}^T, \quad \mathbf{T}^\perp \neq \mathbf{1} - \mathbf{t}\mathbf{t}^T, \tag{3.5}$$

but must rather be defined as block diagonal matrices of projectors of $\mathbb{R}^3$, *i.e.*,

$$\mathbf{T} \equiv \begin{bmatrix} \mathbf{M}_\omega & \mathbf{O} \\ \mathbf{O} & \mathbf{M}_v \end{bmatrix}, \quad \mathbf{T}^\perp \equiv \mathbf{1} - \mathbf{T} = \begin{bmatrix} \mathbf{1} - \mathbf{M}_\omega & \mathbf{O} \\ \mathbf{O} & \mathbf{1} - \mathbf{M}_v \end{bmatrix}, \tag{3.6}$$

where $\mathbf{M}_\omega$ and $\mathbf{M}_v$ are projectors of $\mathbb{R}^3$ defined as in eqs.(3.2) and (3.3) which allow

[1] Array is an ordered arrangement of data elements in one or more dimensions: a list, a table, or a multidimensional arrangement of items.

to project the two vectors, respectively. Finally, equation (3.4) becomes

$$\mathbf{t} = \mathbf{t}_{\mathcal{T}} + \mathbf{t}_{\mathcal{T}}^{\perp} = \mathbf{T}\mathbf{t} + (\mathbf{1} - \mathbf{T})\mathbf{t}. \tag{3.7}$$

3.4 Twist Decomposition Equation

As reviewed in Chapter 2, the relationship between the EE velocity array, called twist and denoted $\mathbf{t}$, and the joint velocities, denoted $\dot{\boldsymbol{\theta}}$, is given by eq.(2.28). The $\mathbf{t}$ and $\dot{\boldsymbol{\theta}}$ are defined as

$$\mathbf{t} \equiv [\boldsymbol{\omega}^T \dot{\mathbf{p}}^T]^T \in 2 \times \mathbb{R}^3, \quad \dot{\boldsymbol{\theta}} \equiv [\dot{\theta}_1 \cdots \dot{\theta}_n]^T \in \mathbb{R}^n, \quad \mathbf{J} \equiv \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \quad \mathbf{A}, \mathbf{B} \in \mathbb{R}^{3 \times n}, \tag{3.8}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity vector of the EE, $\dot{\mathbf{p}} \in \mathbb{R}^3$ is the translation velocity vector of a point of the EE, while $\dot{\theta}_i$ is the velocity of joint i of the manipulator. It is noteworthy that $\mathbf{t}$ is not defined as a vector of $\mathbb{R}^6$, but rather as a set of two vectors of $\mathbb{R}^3$ casted into a column array, and hence, $\mathbf{t} \in 2 \times \mathbb{R}^3 \neq \mathbb{R}^6$.

The inverse kinematic minimum-norm solution of redundant manipulator is given in eq.(2.36). Because it is possible to decompose the twist of the EE into two orthogonal parts for the functional redundant tasks, one lying into task subspace and another one lying into the redundant subspace. Upon substitution of eq.(3.7) into eq.(2.36), we have

$$\dot{\boldsymbol{\theta}} = \underbrace{(\mathbf{J}^{\dagger}\mathbf{T})\mathbf{t}}_{\text{task displacement}} + \underbrace{\mathbf{J}^{\dagger}(\mathbf{1} - \mathbf{T})\mathbf{J}\mathbf{h}}_{\text{redundant displacement}}, \tag{3.9}$$

where $\mathbf{h}$ is an arbitrary vector of $\mathcal{J}$ allowing to satisfy a secondary task. The first part of the right side of eq.(3.9) reaches the joint displacement required by task, while the second part reaches the joint displacement in the redundant subspace.

Twist projector: $\mathbf{T} \equiv \begin{bmatrix} \mathbf{M}_\omega & \mathbf{O} \\ \mathbf{O} & \mathbf{M}_v \end{bmatrix}$		Projector of angular velocity vector: $\mathbf{M}_\omega$			
		0-D task: $\mathbf{O}$	1-D task: $\mathbf{e}\mathbf{e}^T$	2-D task: $\mathbf{1} - \mathbf{e}\mathbf{e}^T$	3-D task: $\mathbf{1}$
Projector of translation velocity vector $\mathbf{M}_v$	0-D task $\mathbf{O}$	$\begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$ All rotation and translation of EE are irrelevant motion.	$\begin{bmatrix} \mathbf{e}\mathbf{e}^T & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$	$\begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$
	1-D task $\mathbf{f}\mathbf{f}^T$	$\begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{f}\mathbf{f}^T \end{bmatrix}$	$\begin{bmatrix} \mathbf{e}\mathbf{e}^T & \mathbf{O} \\ \mathbf{O} & \mathbf{f}\mathbf{f}^T \end{bmatrix}$	$\begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & \mathbf{f}\mathbf{f}^T \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{f}\mathbf{f}^T \end{bmatrix}$
	2-D task $\mathbf{1} - \mathbf{f}\mathbf{f}^T$	$\begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & (\mathbf{1} - \mathbf{f}\mathbf{f}^T) \end{bmatrix}$	$\begin{bmatrix} \mathbf{e}\mathbf{e}^T & \mathbf{O} \\ \mathbf{O} & (\mathbf{1} - \mathbf{f}\mathbf{f}^T) \end{bmatrix}$	$\begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & (\mathbf{1} - \mathbf{f}\mathbf{f}^T) \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & (\mathbf{1} - \mathbf{f}\mathbf{f}^T) \end{bmatrix}$
	3-D task $\mathbf{1}$	$\begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$ Task requiring only 3-D positioning. Orienting are irrelevant Example: gluing	$\begin{bmatrix} \mathbf{e}\mathbf{e}^T & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$ Task requiring 3-D positioning and 2-D orienting. Example: arc-welding; laser-cutting.	$\begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$ All rotation and translation are required. No functional redundancy exist.

Table 3.1 Twist projector matrices.

Equation (3.9) is the main original contribution of this thesis, it does not require the projection onto the nullspace of $\mathbf{J}$ as most of the redundancy-resolution algorithm do, but rather requires an orthogonal projection based on the instantaneous geometry of the task to accomplish.

By defining suitable $\mathbf{M}_\omega$ and $\mathbf{M}_v$, the twist of EE can be projected into 3-dimension, 2-dimension, 1-dimension, or zero-dimension subspace, respectively. From eqs.(3.2) and (3.3), the line projectors of $\mathbf{M}_\omega$ and $\mathbf{M}_v$ are defined as $\mathbf{L}_\omega$ and $\mathbf{L}_v$, respectively, and

$$\mathbf{L}_\omega \equiv \mathbf{e}\mathbf{e}^T; \quad \mathbf{L}_v \equiv \mathbf{f}\mathbf{f}^T. \tag{3.10}$$

Similarly as $\mathbf{e}$, $\mathbf{f}$ is a unit vector along the line $\mathcal{L}_v$ and normal to plane $\mathcal{P}_v$. The plane projectors corresponding to $\mathbf{L}_\omega$, $\mathbf{L}_v$ are

$$\mathbf{P}_\omega \equiv \mathbf{1} - \mathbf{e}\mathbf{e}^T; \quad \mathbf{P}_v \equiv \mathbf{1} - \mathbf{f}\mathbf{f}^T. \tag{3.11}$$

Because both of $\mathbf{M}_\omega$ and $\mathbf{M}_v$ have four possibilities in $\mathbb{R}^3$, the twist projector $\mathbf{T}$ has 16 possibilities as shown in Table 3.1.

According to the requirement of the task, the different twist projector will be selected to decompose the twist. For example, the arc-welding, laser cutting, and milling tasks in Figs. 4.10 and 1.8 have a symmetry EE, and the EE rotation around the EE symmetry axis are irrelevant to the completion of the tasks, so they are tasks requiring 3-D positioning and 2-D orienting, and we can select the twist projector in the fourth line and third column of Table 3.1, *i.e.*,

$$\mathbf{T} = \begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}, \tag{3.12}$$

to decompose twist of EE, where $\mathbf{e}$ is unit vector along the symmetry axis of EE. Arc-welding application will be further discussed in Chapter 4.

In the case of gluing, rotation of EE is irrelevant motion for the task, we can treat it as a task requiring 3-D positioning and 0-D orienting, and hence, the twist projector in the fourth line and the first column of Table 3.1, *i.e.*,

$$\mathbf{T} = \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}, \tag{3.13}$$

could be selected to decompose twist of EE.

3.5 Twist Decomposition Algorithm

Algorithm 3.1 shows the functional redundancy-resolution algorithm with twist decomposition approach. The first step of this algorithm is to decide the required task displacement degree in tool frame according to the task requirement, then the corresponding vector projectors of translation velocities and angular velocities, $\mathbf{M}_\omega$ and $\mathbf{M}_v$ can be selected from Table 3.1, respectively. The following process use the resolved motion rate method shown as Algorithm 2.2, the only difference is Algorithm 3.1 need to computed the unit vectors of irrelevant rotation and translation axes, *i.e.*, $\mathbf{e}$ and $\mathbf{f}$, then substitute them into vector projectors to generate the twist projector $\mathbf{T}$. Instead of using eq.(2.37) as other RR schemes, equation (3.9) is used to solve the IKP and reach $\Delta\boldsymbol{\theta}$ in step 3 of Algorithm 3.1.

3.6 Conclusion

In this chapter, the twist decomposition method is developed to solve the IKP of functionally-redundant serial manipulator. This method orthogonally decompose

Algorithm 3.1: Functional redundancy-resolution algorithm with twist decomposition method

1 Required task displacement degree in tool frame

according the task requirement:

$k-$D translation subtask;

$l-$D orientation subtask;

Select vector projector from Table 3.1:

$\mathbf{M}_v \longleftarrow k$ -D translation velocities vector projector;

$\mathbf{M}_\omega \longleftarrow l$ -D angular velocities vector projector;

$\{\mathbf{p}_d, \mathbf{Q}_d\} \longleftarrow$ desired pose of the EE

$\boldsymbol{\theta} \longleftarrow$ initial guess

2 $\{\mathbf{p}, \mathbf{Q}\} \Leftarrow \mathbf{DKP}(\boldsymbol{\theta})$

$\Delta\mathbf{Q} \Leftarrow \mathbf{Q}^T\mathbf{Q}_d$

$\Delta\mathbf{p} \Leftarrow \mathbf{p}_d - \mathbf{p}$

$$\Delta\mathbf{x} \leftarrow \begin{bmatrix} \mathbf{Qvect}(\Delta\mathbf{Q}) \\ \Delta\mathbf{p} \end{bmatrix}$$

$$\mathbf{DKP}(\boldsymbol{\theta}) \Rightarrow \begin{cases} \mathbf{e} \Rightarrow \mathbf{M}_\omega \\ \mathbf{f} \Rightarrow \mathbf{M}_v \\ \mathbf{J} \end{cases},$$

$$\mathbf{T} \Leftarrow \begin{bmatrix} \mathbf{M}_\omega & \mathbf{O} \\ \mathbf{O} & \mathbf{M}_v \end{bmatrix},$$

3 $\Delta\boldsymbol{\theta} \Leftarrow \mathbf{J}^\dagger\mathbf{T}\Delta\mathbf{x} + \mathbf{J}^\dagger(\mathbf{1} - \mathbf{T})\mathbf{Jh}$

if $\|\Delta\boldsymbol{\theta}\| < \epsilon$ then stop;

else

$\boldsymbol{\theta} \Leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

goto 2

the twist of the EE into a task subspace and a redundant subspace by twist projector, instead of using the null-subspace of Jacobian as other method. Depending on the different task requirement, there are 16 possibilities of the twist projectors. They are defined and discussed in Table 3.1. In this chapter, the functional redundancy-resolution algorithm based on the twist decomposition method is also developed, the algorithm will be applied to solving functional redundant task in the case of arc-welding in next chapter.

CHAPITRE 4

APPLICATION TO ARC-WELDING

4.1 Introduction

In this chapter, the performance of RR schemes are studied in the context when the functional-redundancy is associated to the arc-welding operation. Both the virtual joint and the twist decomposition methods are compared for avoiding the joint limits of the manipulator. Moreover, a discussion is made on the numerical conditioning issue while solving linear algebraic systems. In the following chapters, the kinematic architecture of the serial manipulator PUMA 560 is used to compare and validate the two methods.

The Denavit-Hartenberg parameters of the PUMA 560 are described in Table 4.1.

joint	θ_i	a_i	b_i	α_i
1	θ_1	0.0	0.0	$-\pi/2$
2	θ_2	0.4318	0.0	0.0
3	θ_3	-0.0203	0.1491	$\pi/2$
4	θ_4	0.0	0.4330	$-\pi/2$
5	θ_5	0.0	0.0	$\pi/2$
6	θ_6	0.0	0.055	0
unit	*rad.*	*m*	*m*	*rad.*

Table 4.1 The Denavit-Hartenberg parameters of the Puma 560

4.2 Numerical Conditioning

Every RR schemes must address the numerical conditioning issue when using eq.(2.37) for kinematic redundancy.

In the LAMAT[44] algorithm, the pseudo inverse of the Jacobian is computed directly as:

$$\mathbf{J}^{\dagger} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}, \tag{4.1}$$

where it is apparent that the inverse of $\mathbf{J}\mathbf{J}^{T}$ require to square the condition number of $\mathbf{J}$ before computing the inverse. If the manipulator is closing to a singularity, the condition number of $\mathbf{J}$ becomes greater, and hence, the condition number of $\mathbf{J}^{\dagger}$ is much greater than the one of $\mathbf{J}$. As a consequence, the round-off error of the algorithm is also amplified greatly (Salisbury, Craig[14]), and the algorithm becomes more sensitive to the noise and round-off errors (Angeles[45]).

The relative accuracy of the algorithms becomes clear as the condition number increasing. For a better comparison of the errors created by the algorithms, the manipulator is conducted along a nearby singularity trajectory. Although below the LAMAT algorithm is compared with respect to the AA householder reflection algorithm applied to intrinsic-redundancy, it is also applicable to functional-redundancy.

A MATLAB program which conducts the scheme of Fig. 4.1 is developed. In the program, a closed-singularity posture of PUMA 560 is obtained. According to the desired joint rate, the desired position $\mathbf{p}_d$ and orientation $\mathbf{R}_d$ of the EE is reached by solving a **DKP** function. Since the Jacobian matrix corresponding to the posture can be computed, we can use the two algorithms respectively to solve the IKP of eq.(2.28). After the joints posture are reached , the **DKP** function is used again to calculate the corresponding EE pose. Then, the orientation errors and position

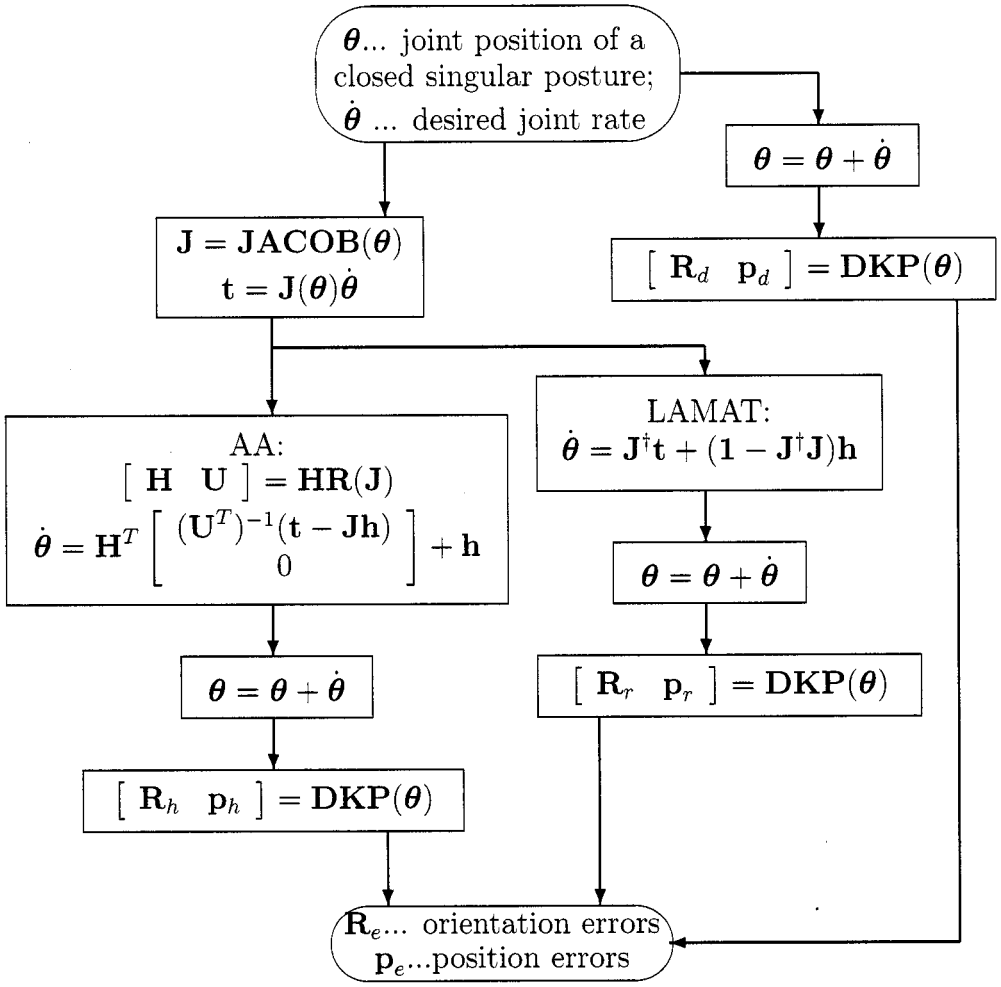


Figure 4.1 The MATLAB program to compare the LAMAT and AA algorithms in solving an ill-condition trajectory

errors of EE can be analyzed.

In both algorithms, the position error is computed as

$$e_p = \|\mathbf{p}_r - \mathbf{p}_d\|, \tag{4.2}$$

where

- $\mathbf{p}_r$ – real position of the EE;

- $\mathbf{p}_d$ – desired position of the EE;
- e_p – position error of the EE;

The orientation error is computed as

$$e_e = \|\mathbf{vect}(\mathbf{Q}_r^T \mathbf{Q}_d)\|; \tag{4.3}$$

where

- e_e – orientation error of the EE;
- $\mathbf{Q}_r$ – real orientation of the EE;
- $\mathbf{Q}_d$ – desired orientation of the EE;
- **vect** – function for transferring the 3×3 matrix to a vector.

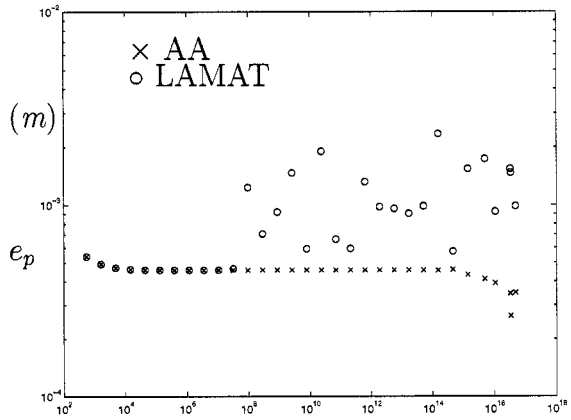


Figure 4.2 Position error of EE

The orientation and position errors are the output of the scheme of Fig. 4.1. Since **J** has both units of radian and length, therefore the use of *millimeter* and *meter* should influence the numerical conditioning of **J**. The millimeter and meter are used in our program respectively for studying the influence of a unit on accuracy.

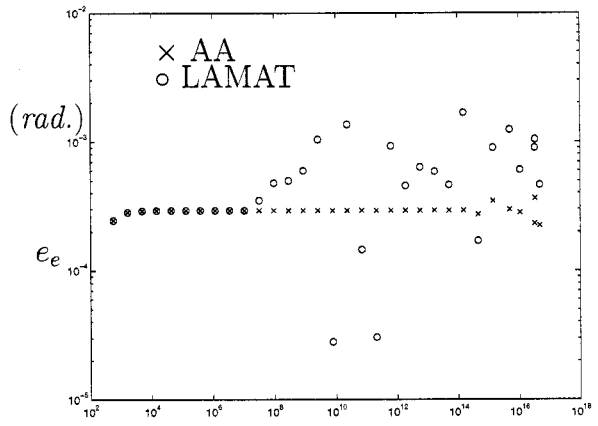


Figure 4.3 Orientation error of EE

Unit	e_p	e_e
Meter	$10^{-4} \sim 10^{-2}$	$10^{-5} \sim 10^{-2}$
Millimeter	$10^{-1} \sim 10^{3}$	$10^{-6} \sim 1$

Table 4.2 Comparison in terms of the error's scatter range and units; e_p— position error; e_e— orientation error on the orthogonal plane of EE

Figures 4.2 and 4.3 respectively show the comparison of position and orientation errors of the EE between the two algorithms, as the unit of the manipulator is in *meter*. We can see the accuracy of two algorithms have great differences when the condition number of the Jacobian **J** is between 10^7 and 10^{14}. AA algorithm maintains the same accuracy, while the accuracy of the LAMAT algorithms scatters in a certain range. For the LAMAT algorithm, the scatter range is also influenced by the unit. For example, the position error is scattered from 10^{-4} to 10^{-2} as the meter is used, and is scattered from 10^{-1} to 10^3 as only changing the unit to millimeter. The scatter range in millimeter is wider than the range in meter as all other parameters are keeping same value, or we can say, meter is more precise than millimeter in this case. Table 4.2 shows the comparison in terms of the error's scatter range and units. Apparently, the use of millimeter produces large errors than meters.

From the numerical comparison between AA and LAMAT algorithm, one concludes the following results:

- the AA householder reflection algorithm can reach a more accurate solution than the LAMAT generalized inverse algorithm, as the manipulator moves closer and closer to a singular posture.
- units affect the scatter range of the result of the LAMAT algorithm. From our experimentation, we observe the use of *meter* in **J** is more accurate than *millimeter* for the size of the PUMA 560 manipulator.

4.3 Pipe-Bride Welding Task

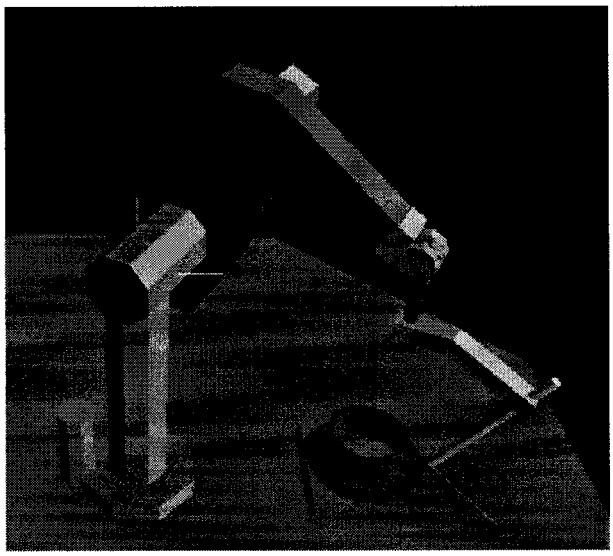


Figure 4.4 The graphic simulator of the welding task implmented by PUMA 560

As shown in Fig. 4.4, a welding tool with transformation matrix $\mathbf{A}_{tool}$ as eq.(4.4)

is installed at the end of PUMA manipulator, and

$$\mathbf{A}_{tool} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\frac{\pi}{6}) & -\sin(\frac{\pi}{6}) & 0.1 \\ 0 & \sin(\frac{\pi}{6}) & \cos(\frac{\pi}{6}) & 0.501 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.4}$$

The PUMA 560 aims to perform a pipe-to-bride welding task, which requiring to perform several consecutive circular trajectories with a period of each turn being: $T = 285$ seconds as given by

$$\mathbf{p} = \begin{bmatrix} 0.1\cos(\omega t) \\ 0.6 + 0.1\sin(\omega t) \\ -0.59 \end{bmatrix}, \tag{4.5}$$

$$\mathbf{Q} = \begin{bmatrix} \cos\alpha & -\sin\alpha\cos\beta & \sin\alpha\sin\beta \\ \sin\alpha & \cos\alpha\cos\beta & \cos\alpha\cos\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix},$$

with

$$\begin{aligned} \alpha &= \tfrac{\pi}{2} + \omega t, \quad \beta = \tfrac{-3\pi}{4}, \\ \omega &= \tfrac{2\pi}{T}, \quad 0 \le t \le T, \end{aligned} \tag{4.6}$$

where distances and angles are, respectively, expressed in meter and radian.

In order to create a continuous trajectory in joint space, the trajectory in task space is sampled at a set of discrete closed poses, and the IKP is solved at each sample pose with Algorithm 2.2. Three different IKP solution methods are used, *i.e.*: no-RR method; virtual joint method; and twist decomposition method.

4.4 No Redundancy-Resolution Method

If we do not take advantage of the symmetry axis of the electrode, arc-welding task implemented by six joint robot is not a redundant task, and hence, equation(2.29) can be directly used to resolve the IKP at each sample pose. As shown in Fig. 4.5, the manipulator is able to perform the task while using the full amplitude of its joint displacement at the first turn. However, a second consecutive turn is, obviously, not possible without exceeding the joint limits.

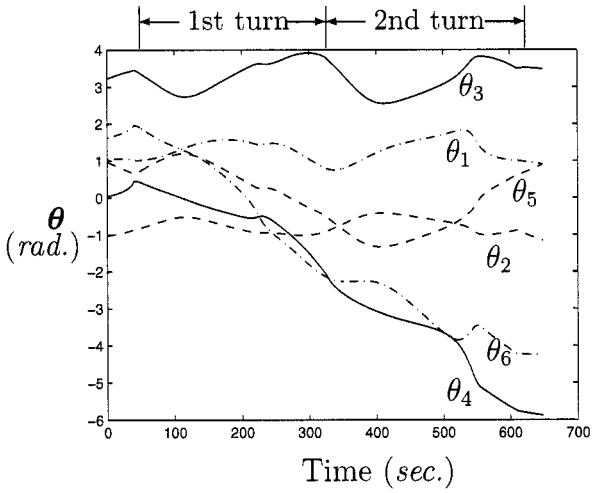


Figure 4.5 Joint position with respect to time without using RR scheme

4.5 Virtual Joint Method

Both the virtual joint method and the twist decomposition method take advantage of the symmetry axis of electrode as an irrelevant motion, and the arc-welding task becomes a functionally-redundant task. Hence, functional RR method can be used to solve the IKP at each sampled pose by using Algorithm 2.2.

In order to better compare the two methods, they must be implemented under

the same conditions, including initial joint position, mean-joint posture, weighted vector, and number of iteration. Here, we select the corresponding initial joint position $\boldsymbol{\theta}_s$ as

$$\boldsymbol{\theta}_s = \begin{bmatrix} \pi/3 & -\pi/3 & \pi & 0 & \pi/3 & \pi/2 \end{bmatrix}^T, \tag{4.7}$$

the mean-joint position $\boldsymbol{\theta}_0$ as

$$\boldsymbol{\theta}_0 = \begin{bmatrix} \pi/2 & -\pi/3 & \pi & \pi/4 & \pi/3 & \pi \end{bmatrix}^T, \tag{4.8}$$

and the weighted vector $\mathbf{w}_v$ as

$$\mathbf{w}_v = \begin{bmatrix} 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 \end{bmatrix}. \tag{4.9}$$

The weighted matrix $\mathbf{W}$ is simply the diagonal matrix of vector $\mathbf{w}_v$. The number of iteration in Algorithm 2.2 is set to 20.

In virtual joint method, a virtual revolute joint around the symmetric axis of the electrode is assumed to be the 7th joint, and the arc-welding task performed by a 6-DOF robot becomes an intrinsically-redundant task. Thus, the various RR algorithms for intrinsic-redundancy can be used on this augmented equivalent manipulator. Because AA householder reflection algorithm can reach a more accurate solution than LAMAT generalized inverse algorithm, equation (2.58) is used in step 2 of Algorithm 2.2. Figure 4.6 shows the time history of the joint positions of the robot for two consecutive turns along the given welding trajectory with virtual joint method. Apparently, the robot is able to perform the task for multiple consecutive turns without exceeding the joint limits. However, excessive joint velocities appear repeatedly at every turn.

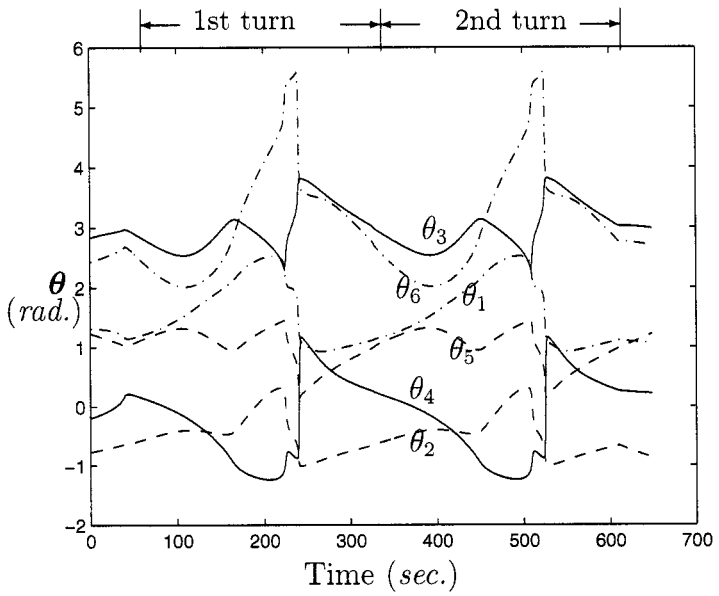


Figure 4.6 Joint positions with respect to time for the virtual joint method

4.6 Twist Decomposition Method

As shown in Table 3.1, if we denote by $\mathbf{e}$ the unit vector along the symmetry axis of the electrode, the twist projector $\mathbf{T}$ onto the arc-welding task is defined as

$$\mathbf{T}_{weld} \equiv \begin{bmatrix} (\mathbf{1} - \mathbf{e}\mathbf{e}^T) & \mathbf{O} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}, \tag{4.10}$$

where

$$\mathbf{e} = \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_5\mathbf{Q}_6\mathbf{v}; \tag{4.11}$$

$$\mathbf{v} \equiv \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T, \tag{4.12}$$

and $\mathbf{1}$ is the (3×3) identity matrix; $\mathbf{Q}_1, ..., \mathbf{Q}_6$ are the rotation matrices from joint 1 to 6. Substituting eq.(4.10) into eq.(3.9) yields

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^\dagger \mathbf{T}_{weld}\mathbf{t} + \mathbf{J}^\dagger \begin{bmatrix} \mathbf{e}\mathbf{e}^T\mathbf{A}\mathbf{h} \\ \mathbf{0} \end{bmatrix}. \tag{4.13}$$

The first part on the right-hand side (RHS) of eq.(4.13) is the arc-welding task, which produces the EE motion on the orthogonal plane to the electrode symmetric axis. The second part is the redundant task, which produces the EE motion around the symmetric axis of the electrode. Equation (4.13) can be used in step 3 of Algorithm 3.1.

Figure 4.7 shows the joint positions for two consecutive turns along the given welding trajectory with the twist decomposition method. Apparently, the manipulator is able to perform multiple consecutive turns without exceeding the joint limits. Excessive joint velocities appears only at the first turn.

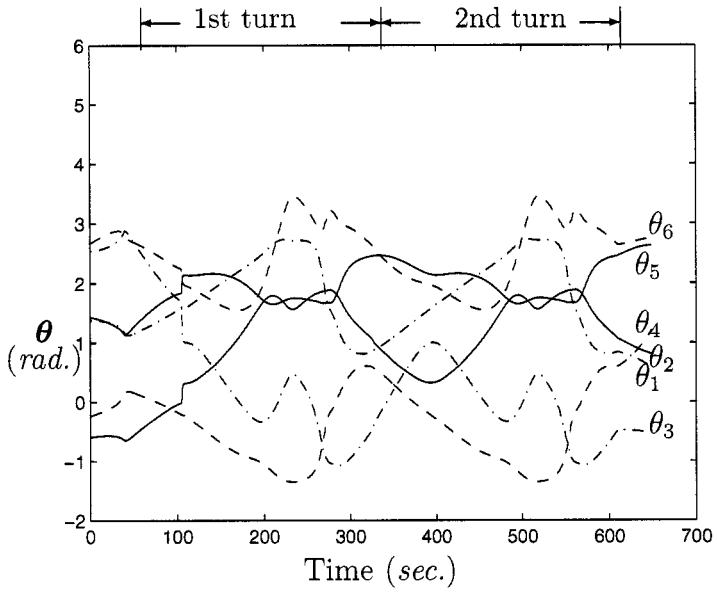


Figure 4.7 Joint positions with respect to time for the twist decomposition method

4.7 Accuracy of Functional RR Methods

In order to compare the accuracy of the two functional RR methods, we study the position and orientation errors between the obtained trajectory of EE with the desired trajectory along the welding path.

If the whole trajectory is composed of n steps, and the position error at step j is the norm of the difference between the reached position $\mathbf{p}_{rj}$ and desired position $\mathbf{p}_{dj}$, *i.e.*,

$$e_{pj} = \|\mathbf{p}_{rj} - \mathbf{p}_{dj}\|, \tag{4.14}$$

then the mean value of the position errors is given as

$$\overline{e}_p = \frac{1}{n}\sum_{j=1}^{n} e_{pj}. \tag{4.15}$$

Similarly, there is an orientation error at step j as the norm of difference between the reached orientation $\mathbf{Q}_{rj}$ and the desired orientation $\mathbf{Q}_{dj}$, *i.e.*,

$$e_{ej} = \|\mathbf{vect}(\mathbf{Q}_{rj}^T\mathbf{Q}_{dj})\|, \tag{4.16}$$

and the mean value of the orientation errors is given as

$$\overline{e}_e = \frac{1}{n}\sum_{j=1}^{n} e_{ej}. \tag{4.17}$$

Since the rotation around the symmetric axis does not affect the welding task, the orientation error along that line is also irrelevant, while only its projection onto the plane normal to that line is meaningful, *i.e.*,

$$o_{ej} = \|(\mathbf{1} - \mathbf{e}\mathbf{e}^T)\mathbf{Q}_{rj}\mathbf{vect}(\mathbf{Q}_{rj}^T\mathbf{Q}_{dj})\|. \tag{4.18}$$

The mean value of array o_e is

$$\overline{o}_e = \frac{1}{n}\sum_{j=1}^{n} o_{ej}. \tag{4.19}$$

It is apparent from Table 4.3 that the twist decomposition method has much lower position and orientation errors in the task space than the virtual joint method.

In other words, the twist decomposition method produces more accurate solutions than the virtual joint method.

method	$\overline{e}_p$	$\overline{e}_e$	$\overline{o}_e$
virtual	0.0789	3.0159×10^{-5}	2.7403×10^{-5}
twist	1.0533×10^{-7}	0.1724	1.0123×10^{-5}
unit	meter	rad.	rad.

Table 4.3 Errors of the virtual joint and twist decomposition methods

However, the twist decomposition method on joint-limits avoiding strategies is more sensitive to both the initial posture setting of the robot and the mean-joint posture, than the virtual joint method. For example, if we change the initial posture for both methods from

$$\boldsymbol{\theta}_a = \begin{bmatrix} \pi/3 & -\pi/3 & \pi & 0 & \pi/3 & \pi/2 \end{bmatrix}^T \tag{4.20}$$

to

$$\boldsymbol{\theta}_b = \begin{bmatrix} \pi/3 & -\pi/2 & \pi/2 & 0 & \pi & \pi \end{bmatrix}^T. \tag{4.21}$$

The implementation of twist decomposition method has completely failed because the joint positions exceed the limits by far, while virtual joint method can still keep the joints within the limits in first turn.

We also notice that there is a moment that the excessive joint velocities occur in the first turn of joint trajectory reached by twist decomposition method. The reason is we don't find the best initial posture for the task. Here, the initial posture was guessed in the trial, it depends on experience and intuitive. How to find a best initial posture is beyond the scope of this thesis. In general, it is more practical to use only the trajectories after the first turn. However, we can not resolve the excessive joint velocities of the virtual joint method, since it is repeated at each turn.

4.8 Conclusion

This chapter has compared two functional RR methods, *i.e.*, the virtual joint method and the twist decomposition method as applied to an arc-welding task with joint-limits avoidance. The virtual joint method assumes a virtual seventh redundant joint for describing the redundant degree of freedom of the task due to the symmetric of the electrode. Since at least one redundant DOF exist, the joint limits avoidance problem can be solved.

Besides the two methods, computational robustness evaluation of two algorithms for intrinsic RR has been conducted on a set of the closed-singular postures. One can summarize the following conclusions:

- the joint space trajectories generated by the twist decomposition method are more reliable than those generated by the virtual joint method in our implementation, *i.e.*, the twist decomposition method can reach a more smooth and more accurate trajectory in joint space;
- twist decomposition method is efficient, *i.e.*, lower computational cost is required by solving a reduced system rather than an augmented one in the virtual joint method;
- the twist decomposition method is more sensitive to the initial posture setting and the weighted vector than the virtual joint method.

CHAPITRE 5

PUMASIM: AN OFF-LINE PROGRAMMING SYSTEM

5.1 Introduction

A robot must be programmed to perform a desired task or motion. There are several methods available for providing an interface between human user and a robot. As previously mentioned, the trajectory in operational space is independent of the robot, and is easily modified to adapt the new task or compensate the implementation error, so it is more practical for trajectory planning of a specific task. However, for precise control of the joint motion of a robot, particularly for redundant tasks such as avoiding obstacles and joint-limits, we need to transform the trajectory from operational space to joint space.

For the purpose of implementing the trajectory in joint space and testing the robot control program before it is executed on the actual robot, an off-line programming and analysis system has been implemented. In this chapter, we present this system called PUMASIM (PUMA SIMulation).

This system allows the operator to modify a task trajectory in operational space with the help of a graphic simulator without the need to write the codes in the robot programming language. Consequently, the skill required for the operators is lower.

This chapter is organized as follows. Section 2 presents an overview of the PUMASIM system. Section 3 describes the preprocessing module which solve the functionally-redundant tasks. Section 4 describes the graphic simulation module

based on XAnimate. Section 5 introduces the VAL II robot programming language and describes the robot language translator module. Section 6 shows an example of PUMASIM. The last section concludes with an indication of the ongoing development of the system.

5.2 Overview of PUMASIM

As shown in Fig. 5.1, PUMASIM comprises the following modules:

1. A *preprocessing module* which generates the trajectory points in joint-space and checks the singularities;
2. A *graphic simulation* module;
3. A *robot-language translator* module;
4. A *communication module* between the robot controller and a micro-computer.

In this system, after the desired EE trajectory in operational space is described, a data file of joint position along the path is created. This preprocessing module, developed in MATLAB, solves the IKP at each EE position and calculates the corresponding configuration of the robot in joint space. Singular poses can be detected and the joint velocities can be calculated through the preprocessing module. Once the trajectory in joint-space is generated, the motion of the manipulator can be displayed graphically with the simulation module, and the trajectory of the manipulator is thus verified to meet our requirements.

If the trajectory satisfies our requirements, a robot dependent language translator generates the corresponding codes of the VAL II program. A VAL II program

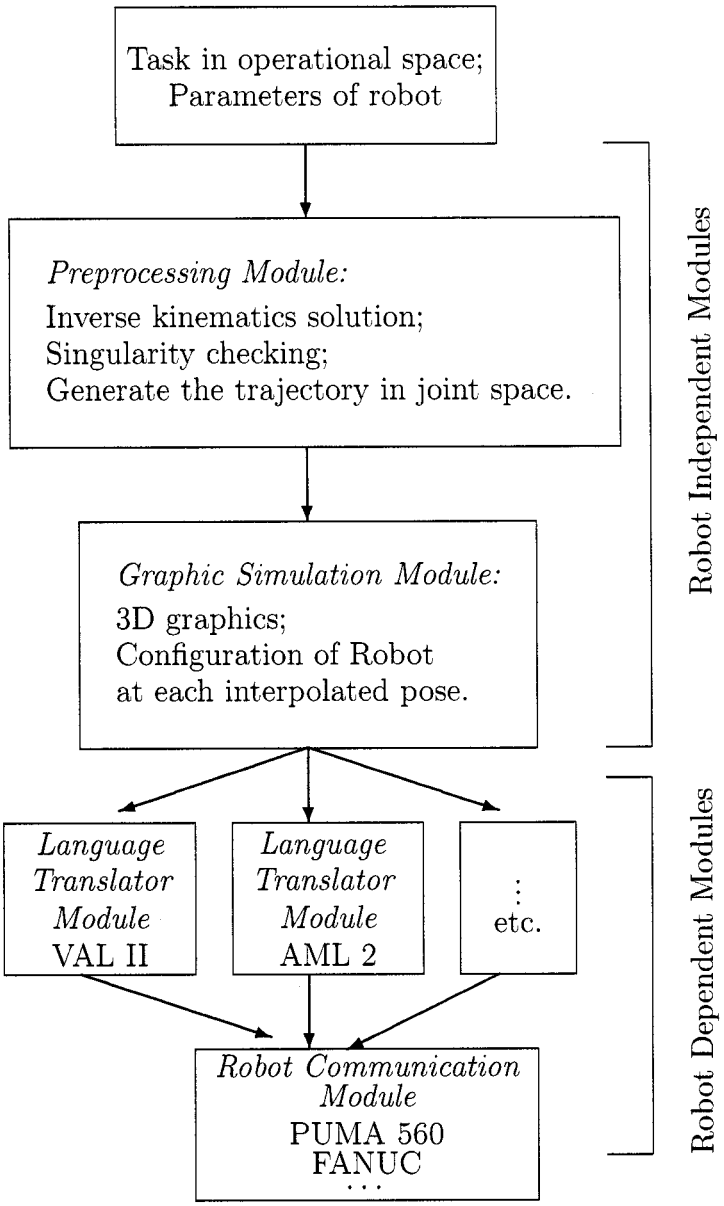


Figure 5.1 The overview of the PUMASIM modules

can be loaded into the controller by serial communication link between the robot controller and the terminal with the communication module. As a result, the task trajectory of the robot is defined in operational space, but the trajectory in joint space is implemented into the robot controller. This integrated robot-programming system has been implemented for the Puma 560, but it can be used with other robots after making some modifications.

5.3 Preprocessing Module

It is an IKP of converting trajectory from EE operational space to joint space. Since an infinite number of solutions to the IKP are available for the redundant tasks, a RR scheme is used to choose one solution from the infinite set.

The preprocessing module is summarized in Fig. 5.2. A file which describes the EE trajectory in operational space is taken as input and a standard output file describing the EE trajectory in joint space is generated.

Linear interpolation, where the source and target poses are joined with a straight line, is used to generate via poses among the defined poses of the desired trajectory in operation space. As these poses are calculated, the joint positions are computed using the inverse kinematics model.

For non-redundant tasks, the inverse of $\mathbf{J}$ is used , *i.e.*,

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{-1}\mathbf{t} \tag{5.1}$$

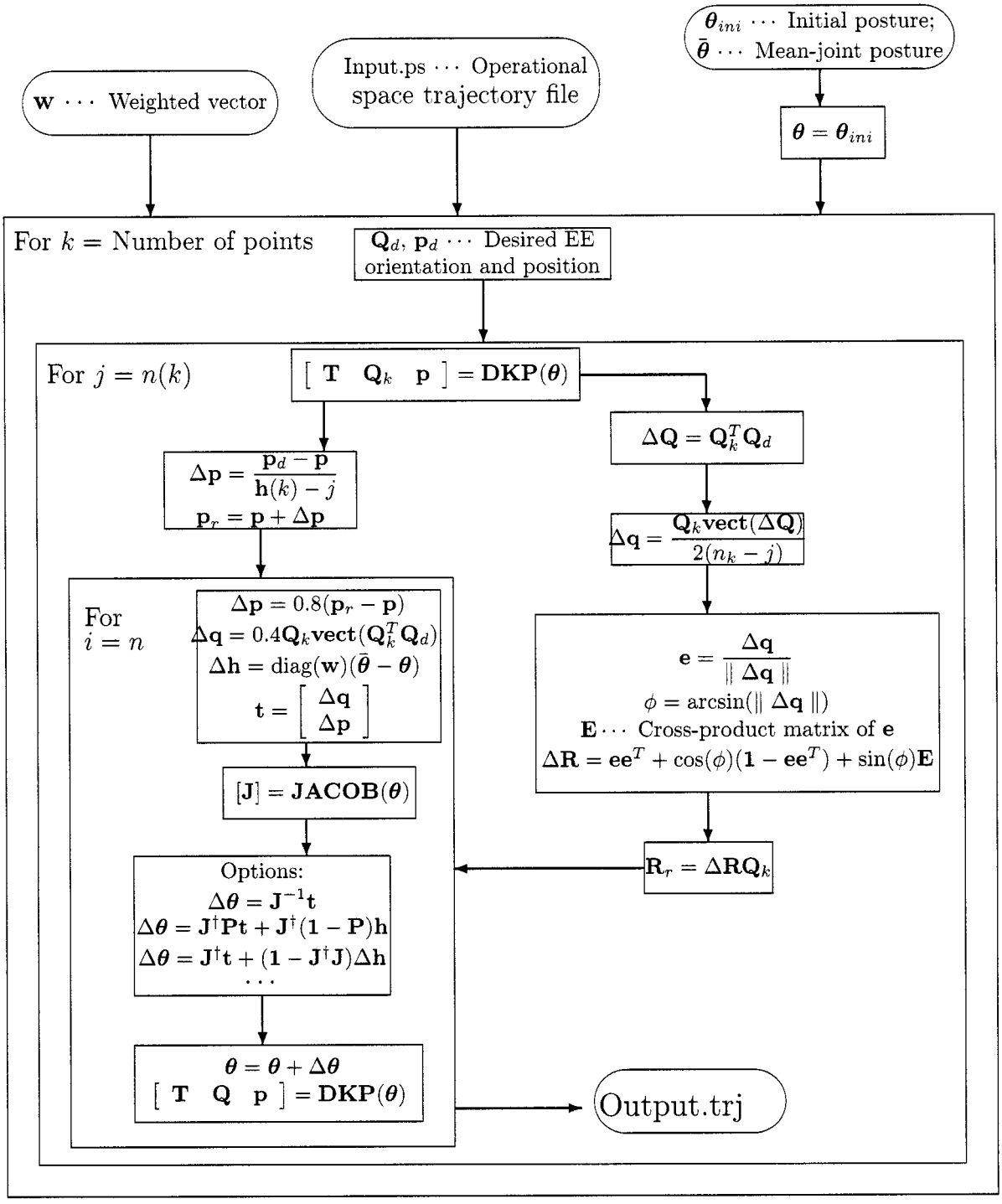


Figure 5.2 Architecture of the preprocessing module

For functionally-redundant tasks, the twist decomposition method is used in the preprocessing module. For intrinsically-redundant tasks, there are several options, such as LAMAT generalized inverse and AA householder reflection algorithms described in Chapter 4.

If there is out-of-joint-limits motion, the preprocessing module stops and provides the position in both operational and joint space. The joint movement limits of Puma 560 is shown as reported in Table 5.1[46]. Singular postures are checked by

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Limits	-2.79 ≀ 2.79	-3.89 ≀ 0.75	-0.89 ≀ 4.05	-1.92 ≀ 2.97	-1.74 ≀ 1.74	-4.64 ≀ 4.64

Table 5.1 Joint displacement limits (in radian) of PUMA 560

using a similar method in the preprocessing module. As a result, the implementation of an out-of-joint-limits or closed-singularity trajectory is avoided by using the preprocessing module.

5.4 Graphic Simulation Module

The graphic simulation module allows to visualize the motions of the manipulator on a computer screen in order to detect and trap any error within a proposed trajectory before an actual robot performs it. These errors may include collisions with obstacles within the workspace, the occurrence of singular posture during the course of the desired path and out-of-joint-limits. The graphic simulation module was developed with the XAnimate library(Mrahefka, Orin[47]).

XAnimate is a C++ library designed to allow users to graphically display a 3D perspective view of robotic manipulators in an X Window with a minimal amount

of effort. It is designed to work across any X Window systems supporting Athena widgets (which are included on most UNIX systems, since they are provided at no cost). Objects may be rendered as a wireframe image, or more importantly as solid color objects. The geometric properties of each object in the system are first described in a XAnimate `.dat` file. An object consists of each part of the robot or environment which moves as a unit, such as the robot base, a link or the object that is manipulated. Each object is then drawn in its correct position and orientation by specifying the transformation matrices between the object's coordinate frame and the world coordinate frame of XAnimate. The robot is simulated by updating the transformation matrices between these coordinate frame at a specified refresh rate.

By using XAnimate, users with almost no computer graphics experience can quickly get an application up and running. A simple Graphic User's Interface (GUI) is included, the only task of the users is to define their objects and placing them in the world coordinate frame.

The sequence shown in Table 5.2 has been used to create our GUI of PUMA 560 as shown in Fig. 5.3. The geometric properties of the tool are realized by two objects; one is scaled from predefined common object `obj_cylinder_z.dat`, and the other is constructed by the `joint_tige.dat` file. Hence, we only need to redefine the objects and change the corresponding DH parameters for adapting a new tool.

5.5 Robot-Language Translator Module

5.5.1 VAL II Language

VAL is a computer based control system and language designed specifically for use with industrial robots of Unimation Inc. The newest version is called VAL II.

1. Read the architecture of manipulator from file `*.par`;

   Read the joint trajectory file `*.trj`;

2. Initialize XAnimate and set the observer position, display rate, position of the center of interest and title;

   Create the geometric properties of objects;

   Set the position of each fixed object;

   Compute the homogenous transformation matrices of the moveable objects with DH parameters of the manipulator;

3. Open and update the interactive display by a predefined function: `XAnimate_event_loop()`.

Table 5.2 Sequence of operation in the graphic simulation module made with XAnimate

As a computed based system, VAL II provides the ability to easily define the task which a robot is going to perform, the ability of responding to information from sensory systems and the ability of working in some unpredictable situations or moving frames of reference.

The VAL II robot language is permanently stored as a part of the VAL II system. The format of the VAL II language is similar to that of the general purpose computer language, BASIC. Each line of the VAL code is similar to that of BASIC, with one instruction per line, followed again by a space and any other argument.

There are two ways to specify movement in VAL: joint by joint programming and spatial programming, respectively. Joint by joint programming is accomplished by assigning numerical value to each joint of the robot. In the spatial programming method, a specified pose in space is given in relation to the robot or with respect to the last pose, and the robot moves to that pose. The spatial programming method is easier to be used than the joint by joint method, however, the joint by joint

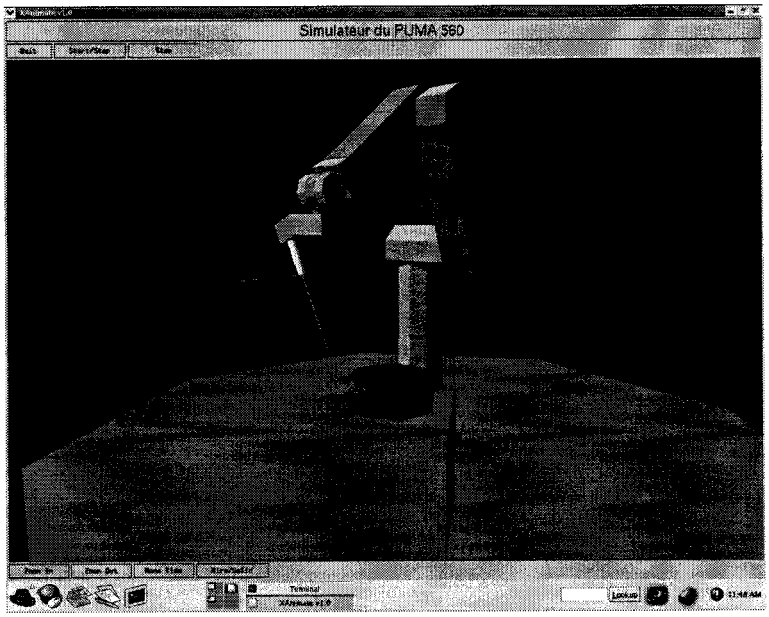


Figure 5.3 Graphical user interface of PUMASIM simulator

method can achieve higher repeatability precision than spatial method[48].

5.5.2 Language Translator Module

The C++ language is used to develop the robot-language translator module. For the purpose of allowing the translator to be used for general programming, a new data type, called `point`, is created in the program. `point` stores each individual joint data related with a specified robot pose. The header file of `point` is as follows:

```
class point {protected:
    double joint1, joint2, joint3, joint4, joint5, joint6;
    int shiftnumber;
public:
    point(double=0.0, double=0.0, double=0.0,
        double=0.0, double=0.0, double=0.0, int=0 );
```

```
                    //default constructor
    point( const point &); //constructor copier
    ~point(){};            //destructor

    void setpoint( double, double , double ,
        double ,double, double);
        //set the rotation degree of every joints
    void setshiftnumber( int & nb);
    int getshiftnumber() {return shiftnumber;}
    void affichepoint(); //print the rotation degree of each joint
    void storepoint(ofstream fic_out);
};
```

We also define an abstract base class: `command`. Each VAL II instruction which has variables is defined as a derived class of `command`. The header file of `command` is as follows:

```
class command {private:
    char *comm;
public:
    command( const char *);
    ~command();

   void storecommand( ofstream fic_out);
   void printcommand();
};
```

There are over 100 instructions in VAL II, but the main robot-controlling instructions of VAL II are:

MOVE: Move the robot gripper to a defined position;

POINT: Define a location variable with a given value;

READY: Return to home position;

APPRO: Approach a location with according to some distance;

DEPART: Back away from a location in accordance to some distance;

SPEED: Set the speed of motion;

SET: Set the location value to the location variable;

PPOINT: Return a precision-point value composed
with the positions of each robot joint;

END: Finish the programming.

The main file of the translator program firstly read as `Input.trj` file, *i.e.*, the trajectory in joint space. Then it stores all the `point` of the file stream into a First-in-First-out (FIFO) linked list. In the list, each `point` is linked with the next `point` until the last `point`. The graphical representation for the list is shown in Fig. 5.4.

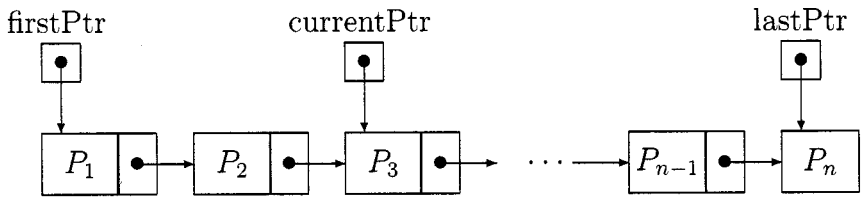


Figure 5.4 Linked list for storing joint position data along the achieved trajectory

After all the poses along the trajectory are defined according to the dates of the list by using class `point`, command `SET #PPOINT` of VAL II are written into `output.v2` file. Following the `READY` and `SPEED` command, the command `MOVE #` is generated according to the poses along the trajectory and written into `output.v2` file. The process model of the Robot-Language Translator module is shown in Fig. 5.5.

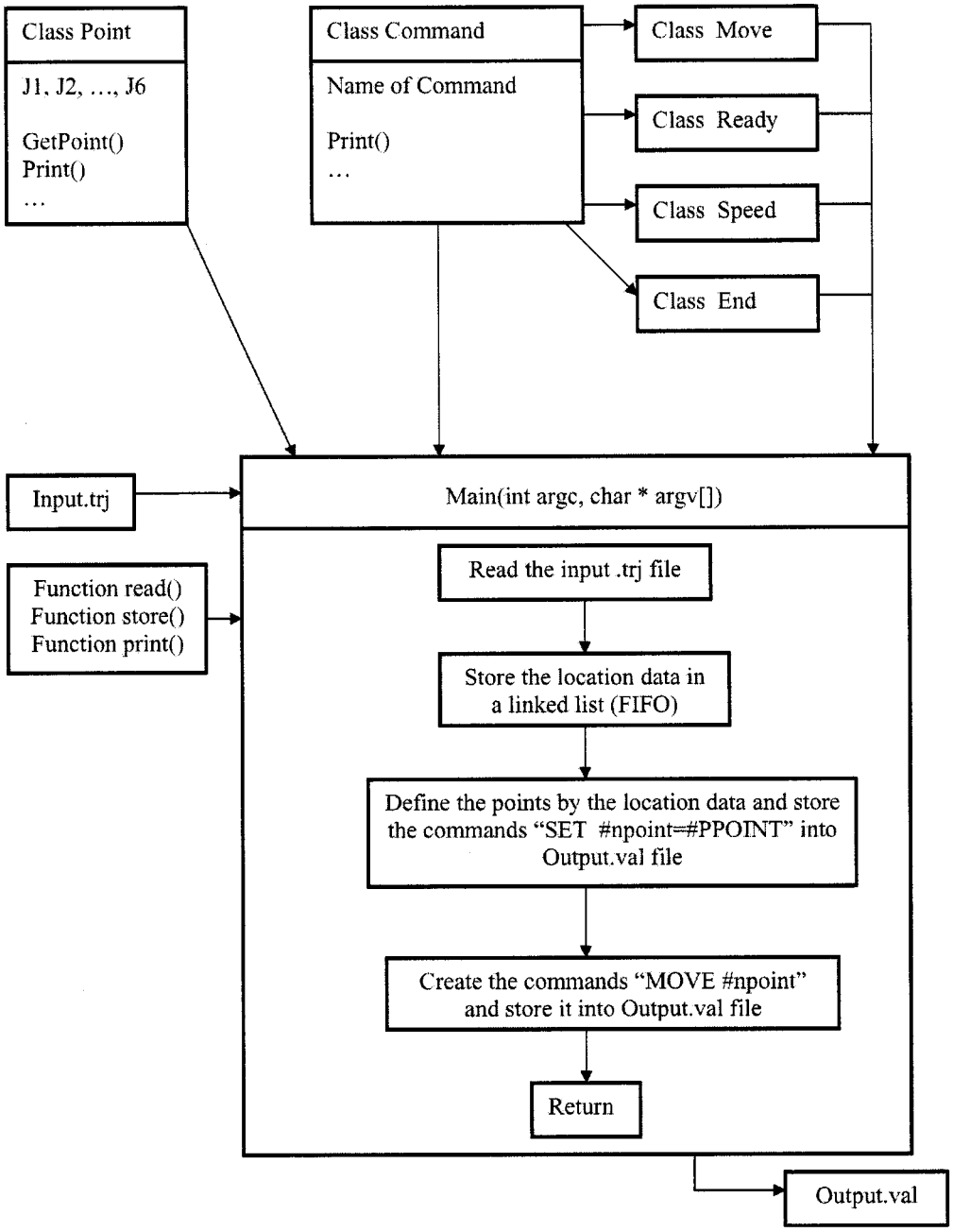


Figure 5.5 Robot-language translator module scheme

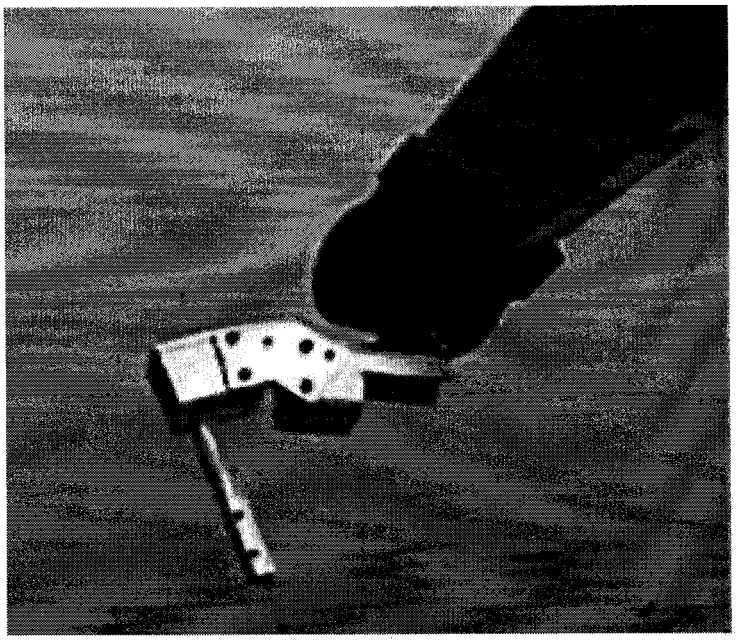


Figure 5.6 Welding tool model

5.6 Demonstration on a PUMA 560 Robot

In order to make a demonstration on a real PUMA 560 robot, a simple welding tool, shown in Fig. 5.6, was designed and fabricated in our machine shop. After measuring the dimension by a Coordinate Measuring Machine (CMM), the tool was installed at the end of the PUMA 560. The DH parameters of Puma 560 has previously been shown in Table 4.1. The transformation matrix of the tool $\mathbf{A}_{tool}$ is

shown as

$$\mathbf{A}_{tool} = \begin{bmatrix} \cos(-0.4398) & 0 & -\sin(-0.4398) & -0.0785 \\ 0 & 1 & 0 & 0 \\ \sin(-0.4398) & 0 & \cos(-0.4398) & 0.154 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.2}$$

Corresponding to the newly installed tool, the preprocessing program and the graphic simulation module of the off-line programming system should be modified accordingly.

Several continuous path demonstrations were implemented to test the system's functions. Now a task similar to the welding task in Chapter 4 is reported here.

5.6.1 A Circular Path with Joint-Limits Problem

The circular path in Cartesian space has a center at point at:

$$[x, y, z] = [0, 500, -100], \tag{5.3}$$

and a radius of 100 mm. Millimeter is the unit of measurement used in Cartesian coordination. The EE must make two consecutive turns along the circular path, while the orientation of EE is constrained to meet:

$$\{R_x, R_y, R_z\} = \{180^o, 15^o, \theta\}, \tag{5.4}$$

with

$$\theta = \omega t, \quad 0 \leq t \leq 180^o. \tag{5.5}$$

There are 180 steps each turn of the circular path.

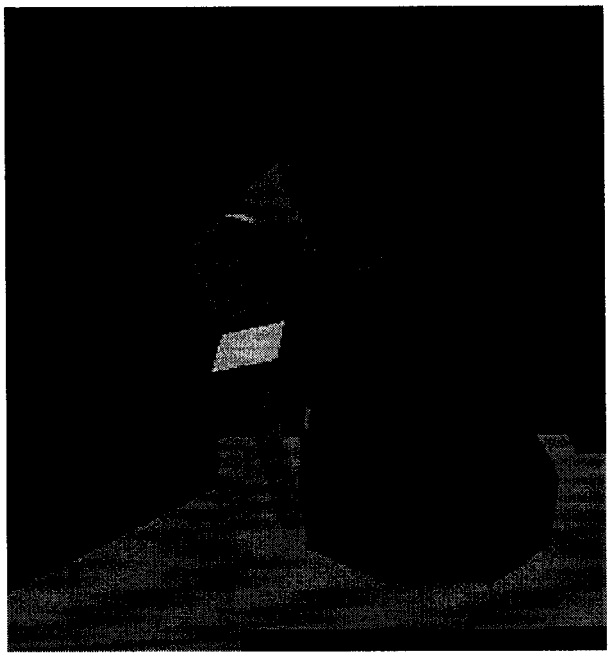


Figure 5.7 A circular trajectory task with joint-limits problem

The task is illustrated in Fig. 5.7. Path planning is done by interpolating a sequence of prescribed path poses, according to defined rates and defined periods among the poses. Hence, the corresponding sequence of values of joint variables is obtained via the IKP solution.

If the defined task was implemented by a non-RR scheme, the fifth joint of the robot is firstly out of the joint limit at the 37th pose. The preprocessing module shows the following message:

```
Joint 5 is out of the limitation, and processing is stopped at
the 37th pose,
the corresponding joint position vector is
[69.7357  -28.8823  148.9392    3.6024   99.9948  184.8341]
```

By ignoring the joint limits checking in the above preprocessing program, the reached joint trajectory without RR scheme is illustrated in Fig. 5.8. Apparently, the fourth joint has exceeded its limit, and the task can not be implemented.

Next, twist decomposition method is used to solve the joint limits problem. With the joint limits and singularity checking in the preprocessing module, the optimized joints trajectories are reached and are shown in Fig. 5.9. Clearly, all the joint positions are within the joint limits and the robot always maintains the good conditioning postures. The corresponding graphic simulation images are illustrated in Fig. 5.10. Corresponding to the optimized joints trajectories, the translator model generates a VAL II program as follows:

```
.PROGRAM  2cir15d
SET #A1=#PPOINT(60,-60,180,0,60,90)
SET #A2=#PPOINT(34.0814,-51.6572,187.025,45.3878,76.6556,-138.227)
SET #A3=#PPOINT(44.6651,-41.0254,161.87,28.5475,92.7235,-169.961)
...
...
...
SET #A1452=#PPOINT(59.5248,-43.471,173.404,9.084,82.7652,-192.255)
SET #A1453=#PPOINT(59.4725,-43.266,173.166,9.180,83.0466,-191.847)

SPEED 20 ALWAYS
READY
MOVE #A1
MOVE #A2
MOVE #A3
...
...
```

```
...
MOVE #A1452
MOVE #A1453
STOP
.END
```

This VAL II program is uploaded into the PUMA 560 controller, and the joints trajectories are implemented successfully as shown in Fig 5.11.

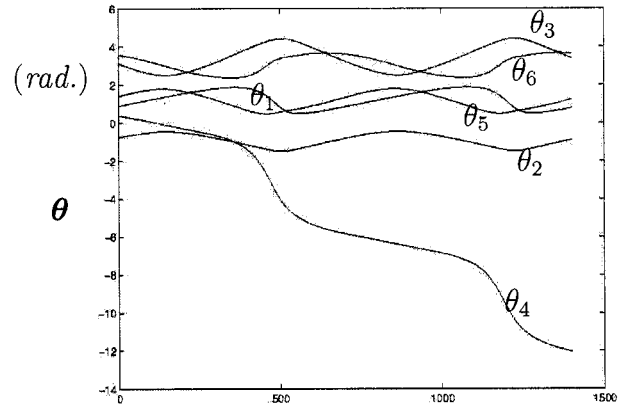


Figure 5.8 Joint positions with respect to time without using RR method

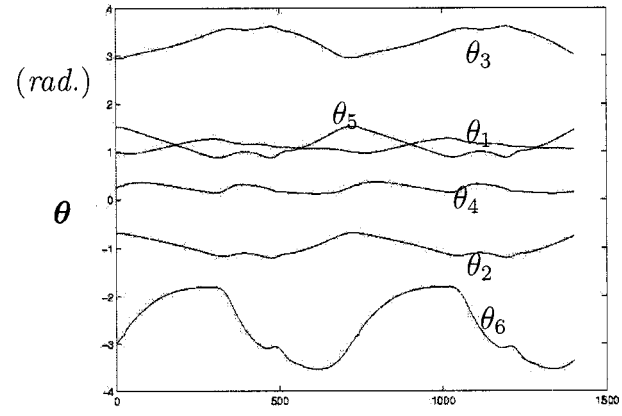


Figure 5.9 Joint positions with respect to time for the twist decomposition method

5.7 Conclusion

PUMASIM provides a high-level user interface to program a task in operational space, which allowing the different methods to propose optimized joint space trajectory to be validated on a graphic simulator, and then, translate the joint space trajectory directly into the robot specific language and executed on a real robot.

PUMASIM has been developed "in-house" using currently available hardware and software devices, *i.e.*, the modification of the joint-rate IKP program to the preprocessing module, the modification of existing graphic simulator to fit new tool and task, the design and the programming of the language translator module, and the using of existing communication module.

Similar commercially available off-line programming packages may be more sophisticated than PUMASIM. However, such "in-house" developed software is very cost-effective and useful for the planning and testing of simple tasks.

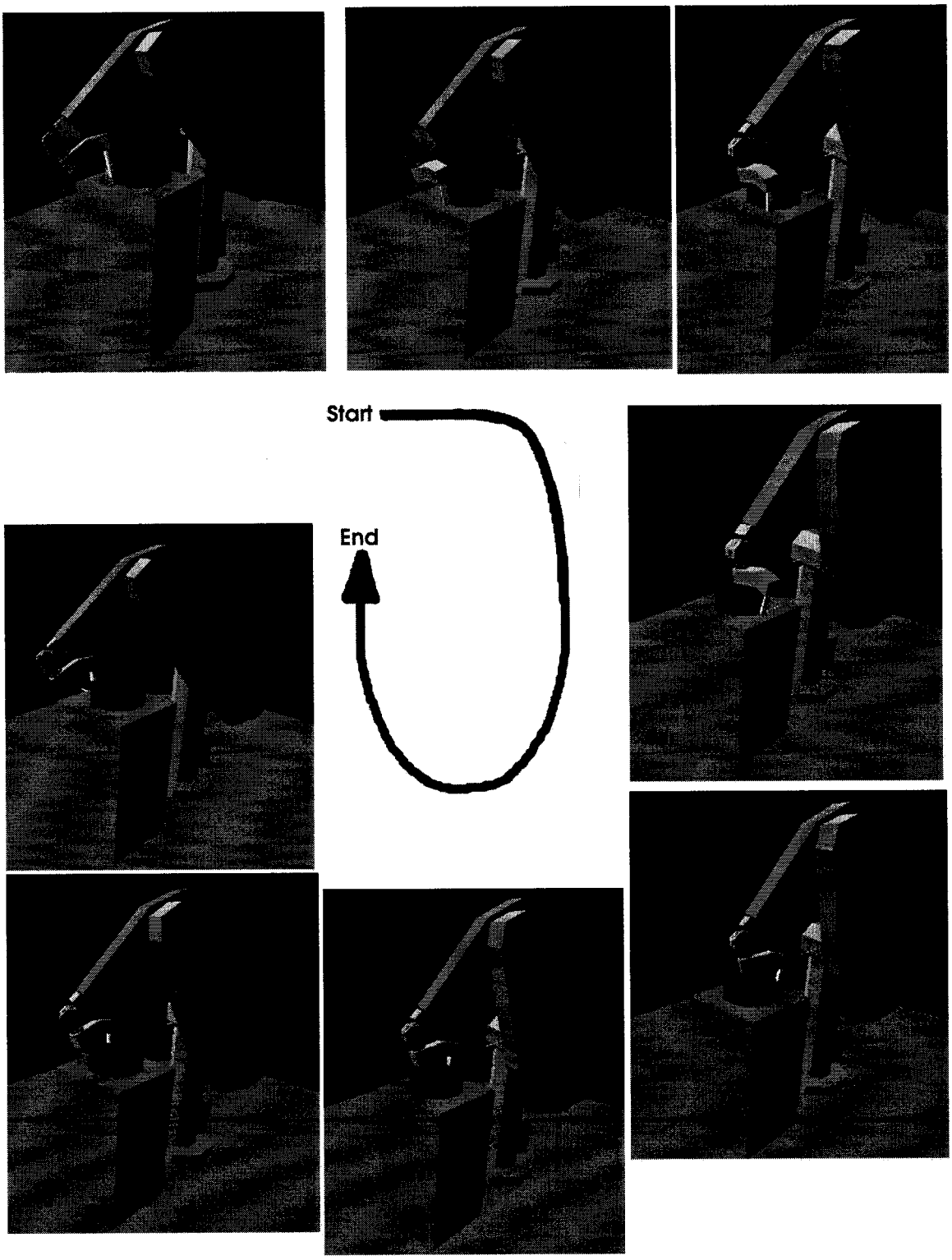
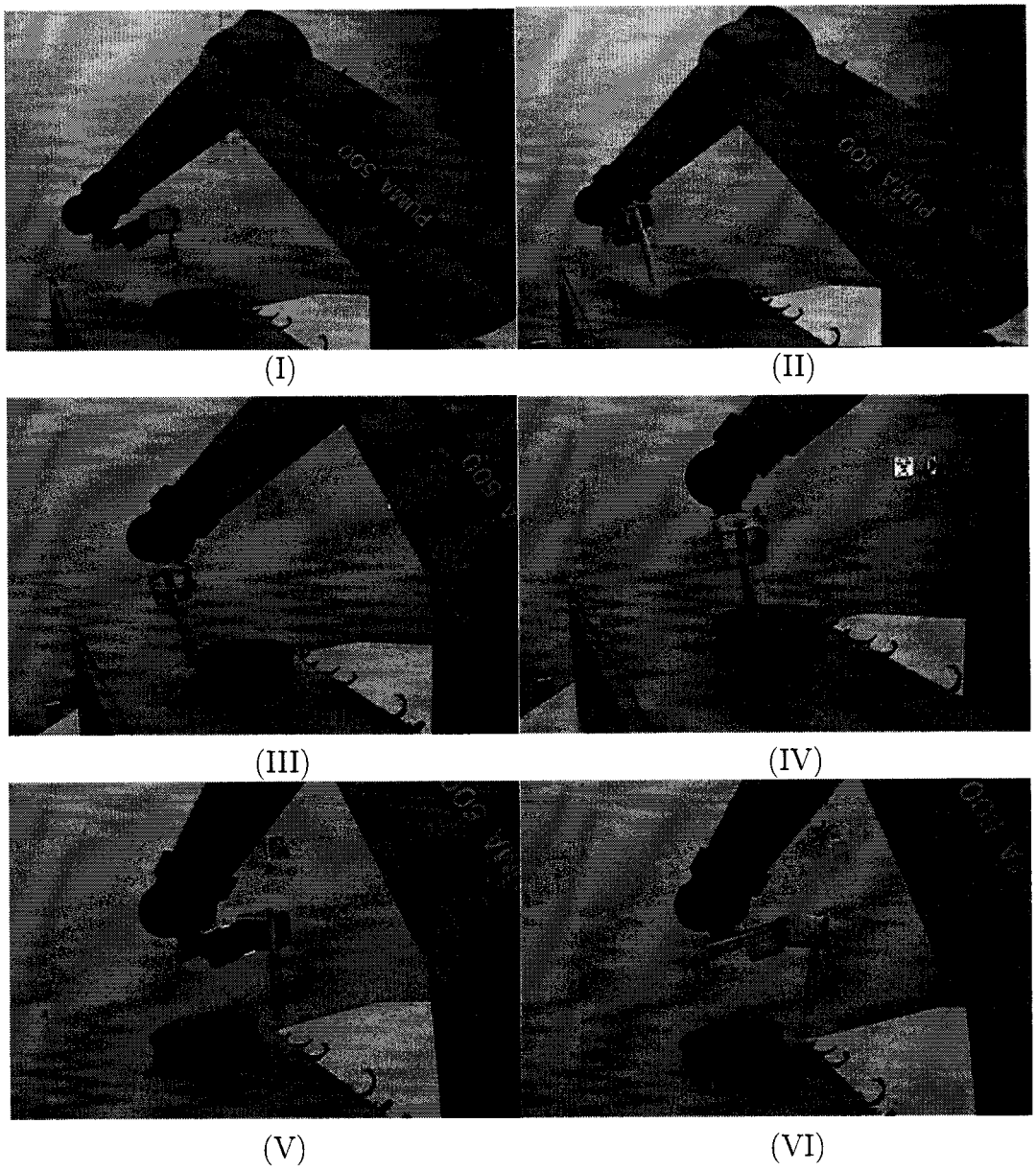


Figure 5.10 The Simulation of the robot trajectory with the twist decomposition method and a joint-limits avoiding strategy



(I) (II)

(III) (IV)

(V) (VI)

Figure 5.11 The implementation on PUMA 560

CHAPITRE 6

CONCLUSIONS

6.1 Original Contribution

In this thesis, the twist decomposition method has been developed for general functionally-redundant manipulators. Instead of projecting on the null space of the Jacobian matrix to optimize the joints trajectories. Twist decomposition method directly decomposes the task into two orthogonal subspaces where the main and secondary tasks lie, respectively. This method has shown to be efficient, *i.e.*, having a low numerical cost, and accurate, *i.e.*: having a low round-off error amplification factor. A numerical example has been shown for arc-welding robotic tasks.

Related to the general RR scheme working on the null space of the Jacobian matrix, this method is also numerically robust such as the AA householder reflection algorithm, which is more precise and stable than the LAMAT generalized inverse algorithm, particularly on closed-singularity postures.

6.2 Future Research Direction

In previous chapters, it is shown that twist decomposition method is able to reach a smoother and more accurate joints trajectories than virtual joint method in our tested task. Although twist decomposition method can only solve functionally-redundant problems, it is still very meaningful and interesting since a high number of functionally-redundant tasks exist in industrial application, such as welding, spraying, milling, water-jet cutting and laser cutting, etc.

In the cases where both intrinsically and functionally-redundant exist together in a kinematic manipulator, the different RR schemes can be combined and used together. For example, supposing a seven revolute joints manipulator implementing a 5-DOF task in 3D operational space, hence, it is a combined kinematic redundant manipulator, there is 1-DOF intrinsic-redundancy and 1-DOF functional-redundancy. Obviously, we can use a non minimum norm general inverse solution as in eq.(2.37) to solve the intrinsically redundant part. After decomposing the task into two components with the twist projector matrix, and substituting the two components into eq.(2.37), the combined redundancy resolution for the manipulator could be written as

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\dagger}\mathbf{T}\mathbf{t} + \mathbf{J}^{\dagger}(\mathbf{1} - \mathbf{T})\mathbf{t} + (\mathbf{1} - \mathbf{J}^{\dagger}\mathbf{J})\mathbf{h}, \tag{6.1}$$

Equation (6.1) is only a possible resolution of combined redundant manipulator, the fully study on combined RR schemes is left as future research subject.

During the process of optimization, we notice that the selection of an initial posture, weighted vector greatly affect the optimized trajectory. A bad selection may even cause the optimization failing completely. Actually, these selections rely mostly on experience. It is necessary to develop a more effective and rational method. In following years, this research subject could integrate the RR solution with intelligent control methods, such as fuzzy logic, neutral network and genetic algorithms. In recent years, some researchers already applied fuzzy logic theory and neural network theory on the control of robotic manipulators [49][50][51].

On the other hand, twist decomposition method is only applied on the joint-limit avoidance problem in this thesis. It requires more study on other applications, such as obstacle avoidance and singularity avoidance.

We also recognize that our application task of arc-welding is a simplified one. In

reality, some welding tasks are much more complicated. For example, in some cases, the rotation along torch symmetric axis are not irrelevant, then the tasks may require the full 6-DOF to realize it.

REFERENCES

[1] Angeles, J., *Fundamentals of robotic mechanical systems: theory, methods and algorithms*, Springer, New York, 521 pages, 2003.

[2] Mitsubishi Heavy Industries, LTD., http://www.sdia.or.jp/mhikobe_e/products/mechatronic/specfi/specfi_e.html.

[3] Laval University, http://wwwrobot.gmc.ulaval.ca/recherche/theme01_a.html.

[4] Laval University, http://wwwrobot.gmc.ulaval.ca/recherche/theme04_a.html.

[5] COMET-II, Chiba University, http://mec2.tm.chiba-u.jp/%7Enonami.

[6] Nakamura, Y., *Advanced robotics redundancy and optimization*, Addison-Wesley Pub. Co., Massachusetts, 337 pages, 1991.

[7] Canadian Space Agency, http://www.space.gc.ca/asc/app/gallery/results2.asp?image_ id=mss_spar2.

[8] Eric W. Weisstein. *Null Space.* MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/NullSpace.html

[9] Denavit, J. and Hartenberg, R.S., *A kinematic notation for lower pair mechanisms based on matrices,* J. Applied Mechanics, vol. 22, pp. 215–221, 1955.

[10] Mooring, W. B., Roth, Z. S. and Driels, M. R., *Fundamentals of manipulator calibration*, A Wiley-interscience Publication, 329 pages, 1991.

[11] Sciavicco, L. and Siciliano, B., *Modelling and control of robot manipulators*, Springer, London, 377 pages, 2000.

[12] Pieper, D.L., *The Kinematics of Manipulators under Computer Control,* Ph.D. thesis, Stanford University, 1968.

[13] Whitney, D.E., *Resolved motion rate control of manipulators and human prostheses.* IEEE Trans. Man-Machine Syst., vol. 10, no. 2, pp. 47–53. 1969.

[14] Salisbury, J. K. and Craig, J.J., *Articulated hands: force control and kinematic issus.* The Int. J. of Robotics Research, Vol. 1, No. 1, pp. 4–17, 1982.

[15] Angeles, J. and Rojas, A., *Manipulator inverse kinematics via condition number minimization and continuation*, International Journal of Robotics and Automation, vol. 2, No. 2, pp. 61–69, 1987.

[16] Golub, G.H. and Van Loan, C.F., *Matrix computations,* The Johns Hopkins University Press, Baltimore and London, 1989.

[17] Angeles, J., Rojas, A. and Lopez-cajun, C.S., *Trajectory planning in robotics continuous-path application*, IEEE J. of Robotics and Automation, Vol. 4. No. 4, pp. 380–385, August, 1988.

[18] Dahlquist, G. and Björck, A., *Numerical Methods,* Prentics-Hall, Englewood Cliffs, New Jersey, 573 pages, 1974.

[19] Whitney, D.E., *The mathematics of coordinated control of prosthetic arms and manipulator,* ASME J. Dynamics Systems, Measurement and Control, Vol. 94, No. 4, pp. 303–309, 1972.

[20] Keramas, J.G., *Robot Technology Fundamentals,* Delmar Publishers, Albany, New York, 407 pages, 1999.

[21] Monsberger, W. and Rapela, D. R., *Rcellsim: a graphic simulaiton system of robot work cells*, Proc. 11th. ISPE/IEE/IFAC Int. Conf. on CAD/CAM, Robotics and Factories of the Future, Pereira, Colombia, Aug. 1995.

[22] Lee, D.M.A. and Elmaragy, W.H., *Robosim:a CAD-based off-line programming and analysis system for robotic manipulators*, Computer-Aider Engineering Journal, Vol. 10, pp. 141–149, Oct. 1990.

[23] Baillieul, J., *Kinematic programming alternative for redundant manipulator,* Proc. International Conference on Robotics and Automation, St-Louis, pp. 722–728, 1985.

[24] Baillieul, J., *Avoiding obstacles and resolving kinematic redundancy,* IEEE International Conference on Robotics and Automation, Washington, pp. 1698–1704, 1986.

[25] Klein, C. A., *Use of redundancy in the design of robotic systems,* 2nd, Int. Symp. of Robotics Research, MIT Press, pp. 207–214, August 1984.

[26] Honegger, M. and Codourey, A., *Redundancy resolution of a cartesian space operated heavy industrial manipulator,* IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2094–2098, 1998.

[27] Chahbaz, A., *Analyse et développement de la méthode de compensation des vitesses articulaires de robots redondants,* M.Ing thesis, École Polytechnique de Montréal, 1994.

[28] Yashi, O.S., and Ozgoren, K., *Minimal joint motion optimization of manipulators with extra degrees of freedom,* Mechanism and Machine Theory, Vol. 19, No. 3, pp. 325–330, 1984.

[29] Angeles, J., Anderson, K. and Gosselin, C., *An Orthogonal-Decomposition Algorithm for Constrained Least-Square Optimization,* ASME Robotics, Mechanisms and Machine Systems, Design Eng. Division, Vol. 2, pp. 215–220, 1987.

[30] Siciliano, B., *Solving manipulator redundancy with the augmented task space methode using the constraint Jacobian transpose,* IEEE Intern. Conf. on Robotics and Automation, Tutorial M1, pp. 5.1–5.8, 1992.

[31] Weisstein, E.W., *Moore-Penrose Matrix Inverse.* MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/Moore-PenroseMatrixInverse.html.

[32] Klein, C. and Huang, C.H., *Review of pseudoinverse control for use with kinematically redundant manipulators,* IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-13, No. 3, pp. 245–250, 1983.

[33] Arenson, N., Angeles, J. and Slutski, L., *Redundancy-resolution algorithms for isotropic robots,* Advances in Robot Kinematics: Analysis and Control, pp. 425–434, 1998.

[34] Liégeois, A., *Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms,* IEEE Trans. Syst., Man, Cybern., vol. SMC-7, pp. 245–250, Mar. 1977.

[35] Yoshikawa, T., *Analyais and control of robot manipulators with redundancy,* Robotics Research: The First International Symposium, pp. 735–747, 1984,

[36] Kosuge, K. and Furuta, K., *Kinematic and dynamic analysis of robot arm,* IEEE Int. Conf. on Robotics and Automation, pp. 1039–1044, 1985.

[37] Hanafusa, H., Yoshikawa, T. and Nakamura, Y., *Analysis and control of articulated robot arms with redundancy,* Prep. 8th IFAC World Congress, XIV, pp. 78–83, Aug. 1981.

[38] Nakamura, Y. and Hanafusa, H., *Inverse kinematic solutions with singularity robustness for robot manipulator control,* ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 108, No. 3, pp. 163–171, 1986.

[39] Kelmar, L. and Khosla, P.K., *Automatic generation of forward and inverse kinematics for a reconfigureable modular manipulator,* Journal of Robotics Systems, Vol. 7, No. 4, pp. 599–619, 1990.

[40] Park, J., Chung, W. and Youm, Y., *Weighted decomposition of kinematics and dynamaics of kinematically redundant manipulators,* IEEE International Conference on Robotics and Automation, Vol. 1, pp. 480–486, 1996.

[41] Chang, T.-F. and Dubey, R.-V., *A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators,* IEEE Trans. Robot. Automat.,vol. 11, pp. 286–292, Apr. 1995.

[42] Baron, L. *A joint-limits avoidance strategy for arc-welding robots,* International Conference on Integrated Design and Manufacturing in Mechanical Engineering, Montreal, Canada, May 2000.

[43] Baron, L. *An Optimal Surfacing Post-Processor Module for 5-Axes CNC Milling Machines,* Third International Conf. on Industrial Automation, Montréal, Canada, June 1999.

[44] Anderson, E., Bai, Z. and C. Bischof, et *al.*, *LAPACK User's Guide,* http://www.netlib.org/lapack/lug/lapack_lug.html, Third Edition, SIAM, Philadelphia, 1999.

[45] Angeles, J., Bulca, F. and Arenson, M., *The kinematic sensitivities of redundant robotic manipulators,* Proc. 5th International Symposium on Advances in Robot Kinematics (ARK), pp. 17–26, 1996.

[46] Unimation Inc. *Unimate PUMA Mark II Robot: 500 series Equipment Manual for VAL II and VAL PLUS Operating Systems*, Mar. 1985.

[47] Marhefka, D.W. and Orin, D.E., *XAnimate v1.0 User's Guide*, Aug. 1995.

[48] Unimation Inc. *Unimate Industrial Robot: Programming Manual User's Guide to VAL II* . Version 2.0, Feb. 1986.

[49] R. Palm, *Control of a redundant manipualtor using fuzzy rules,* Fuzzy Sets and Systems, Vol. 45, No. 3, pp. 279–298, Feb. 1992.

[50] Buckley, K.A., Hopkins, S.H. and Turton, B.C.H., *Solution of kinematics problem of a highly kinematically redundant manipulator using genetic algorithms,*

IEEE, Genetic Algorithms in Engineering Systems: innovations and applications, pp. 264–269, Sep. 1997.

[51] Parikh, P.J. and Lam, S.Y., *A Hybrid Strategy to Solve the Forward Kinematics Problem in Parallel Manipulator,* IEEE Transactions on Robotics, Vol. 21, No. 1, pp. 18–25, Feb. 2005.